Modulo-$(2^n - 2^q - 1)$ Parallel Prefix Addition via Excess-Modulo Encoding of Residues

Seyed Hamed Fatemi Langroudi
Electrical and Computer Engineering Department,
Shahid Beheshti University, Tehran, Iran
s.fatemi.2013@ieee.org

Ghassem Jaberipur[1]
Computer Science & Engineering Department,
Shahid Beheshti University, Tehran, Iran
jaberipur@sbu.ac.ir

Abstract*— The residue number system $\tau = \{2^n - 1, 2^n, 2^n + 1\}$ has been extensively studied towards perfection in realization of efficient parallel prefix modular adders, with $(3 + 2\log n)\Delta G$ latency. Many applications, such as digital signal processing require fast modular operations. However, relying only on τ limits the magnitude of n, and accordingly the dynamic range. Therefore, additional mutually prime moduli are required to accommodate for wider dynamic range. On the other hand, speed of modular arithmetic operations for the additional moduli should be as close as possible to those in τ. This could be best met by the moduli of the form $2^n - (2^q + 1)$, with $1 \le q \le n - 2$, such as $2^n - 3, 2^n - 5$. However, the fastest parallel prefix realization of modulo-$(2^n - 2^q - 1)$ adders that we have encountered in the relevant literature, claims $(7 + 2\log n)\Delta G$ latency. Motivated by the need to reduce the latter, we propose new designs of such adders with $(5 + 2\log n)\Delta G$ latency without any penalty in area consumption or power dissipation. The proposed modular addition algorithm entails supplementary representation of residues in $[0, 2^q]$, as $[2^n - (2^q + 1), 2^n - 1]$. This leads to additional performance efficiency similar to the effect of double zero representation in modulo-$(2^n - 1)$ adders. The aforementioned analytically evaluated speed gain and improvements in other figures of merit are also supported via circuit simulation and synthesis.

Keywords-Residue number system; Parallel prefix modular adder; Excess-modulo encoding;

I. INTRODUCTION

Modular adders are required in many digital applications, such as cryptography [1], and digital signal processing (DSP) via residue number systems (RNS) [2]. Modulo-m adders implement Eqn. 1.

$$|X + Y|_m = \begin{cases} X + Y, & \text{if } X + Y < m \\ X + Y - m, & \text{if } X + Y \ge m \end{cases} \quad (1)$$

Performance of these adders depend, by and large, on the value of m. For example, in case of $m = 2^n$, conventional unsigned binary adders, with alternative architectures (e.g., ripple carry, parallel prefix) can be used. Modulo-$(2^n - 1)$ adders can be also efficiently realized via direct utilization of conventional one's complement adders. This is based on Eqn. 2, at the cost of imposing two representations for zero (i.e., 0 and $2^n - 1$) [3].

[1] G. Jaberipur is also affiliated with School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran.

$$|X + Y|_{2^n - 1} = \begin{cases} X + Y, & \text{if } X + Y < 2^n \\ |X + Y + 1|_{2^n}, & \text{if } X + Y \ge 2^n \end{cases} \quad (2)$$

These adders end-around the carry-out of $X + Y$ and actually compute $|X + Y + \text{EAC}|_{2^n}$, where EAC stands for end-around carry. A regular parallel prefix (RPP) realization that manipulates EAC via an extra parallel prefix level [3], shows $(5 + 2\lceil \log n \rceil)\Delta G$ latency, where ΔG denotes the delay of a simple gate. Although the excellent EAC-fusing technique of [3] removes the extra parallel prefix level, thus reducing the delay to $(3 + 2\lceil \log n \rceil)\Delta G$.

Conventional realization of modulo-m adders, for $m = 2^n - \delta$, and $\delta > 1$, implement Eqn. 3.

$$|X + Y|_{2^n - \delta} = \begin{cases} X + Y, & \text{if } W < 2^n \\ |W|_{2^n}, & \text{if } W \ge 2^n \end{cases}, W = X + Y + \delta \quad (3)$$

This scheme, whose various realizations [4-9] require considerable additional logic besides one n-bit adder, entails some speed loss. Innovative solutions, which are more latency-compatible with the case of $\delta = 1$, have been appeared in the relevant literature; namely a parallel prefix generalized solution for $\delta = 2^q + 1 (1 \le q \le n - 2)$ [10], and one specifically for $q = n - 2$ [11]. Note that $q = n - 1$, is excluded since it leads to modulo $2^{n-1} - 1$. These solutions, unlike to that of Eqn. 2 (for $\delta = 1$), rely on partial or full computation towards obtaining both $X + Y + \delta$ and $X + Y$. The former accommodates two weighted-1 and weighted-2^q EACs of $X + Y + \delta$ within the parallel prefix network (PPN). The latter, however, does it via some shared logic, where the selective carry signal is that of $X + Y + \delta$. Neither are based on EAC addition of $X + Y$, which is however, the case in [12-13] for $q = 1$ (i.e., $\delta = 3$). Therefore, as generalization of the latter EAC-based method, we are motivated to study the design and implementation of modulo-$(2^n - \delta)$ parallel prefix adders that compute $|X + Y + \delta \times \text{EAC}|_{2^n}$, for $\delta = 2^q + 1$. Other encouraging sources for this study are the existence of modulo-$(2^n - \delta)$ residue generators [14-16], and multipliers [17-19]. Moreover, there are cryptosystems [20], and DSP applications [21] that take advantage of such moduli.

The remaining sections of this paper are organized as follows. A background on RNS and modular adders is offered in Section II. Section III provides our general solution for parallel prefix modulo-$(2^n - 2^q - 1)$ adders. Performance evaluation and comparison with previous works, analytically and by synthesis, are taken up in Section IV, and we conclude in Section V.

1063-6889/15 $31.00 © 2015 IEEE
DOI 10.1109/ARITH.2015.9

II. BACKGROUND

Modular adders are vastly used in RNS applications. A typical RNS is defined via a moduli set $\{m_1, \dots m_k\}$, where the k moduli are commonly pair wise prime in order to maximize the cardinality (aka dynamic range) of represented numbers by the RNS in hand. As such, integers in $[0, M)$ are supported by the RNS, where $M = \prod_{i=1}^{k} m_i$.

Addition of two numbers U, and V, both in $[0, M-1]$, is decomposed to k modulo-m_i addition operations $|u_i + v_i|_{m_i}$ for $1 \le i \le k$, in parallel, where $u_i = |U|_{m_i} = U - \lfloor U/m_i \rfloor \times m_i$ is the remainder of integer division U/m_i, and similarly $v_i = |V|_{m_i}$.

RNS is usually beneficial in applications where several additions and multiplications take place before RNS-to-binary conversion is required (e.g., FIR filters [22-23]). Latency of such multi-channel operation is determined by the slowest channel corresponding to one of the moduli. There is usually one modulo-2^n channel, where the corresponding conventional n-bit adder is the fastest with lowest cost.

The next best performance is experienced for modulo-$(2^n - 1)$ adders, where a conventional one's complement adder does the job. However, in case of $u + v = 2^n - 1$, which should normally lead to $|u + v|_{2^n-1} = 0$, the one's complement adder provides $2^n - 1$, as the resulted modular sum. This value is recognized as the second representation for 0 with no problem in correctness of the subsequent operations, since for instance $|u + 2^n - 1|_{2^n-1} = |u + 0|_{2^n-1} = u$, and $|u \times (2^n - 1)|_{2^n-1} = u \times |2^n - 1|_{2^n-1} = 0$.

A. Parallel prefix modulo-$(2^n - 1)$ adders

The modular addition $|X + Y|_{2^n-1}$, of Eqn. 2, with double zero representation, can be also described via Eqn. 4, where $W = w_n w_{n-1} \dots w_0 = X + Y$, and EAC $= w_n$.

$$|X + Y|_{2^n-1} = \begin{Bmatrix} W, & \text{if } W < 2^n \\ |W + 1|_{2^n}, & \text{if } W \ge 2^n \end{Bmatrix} = w_{n-1} \dots w_0 + w_n \quad (4)$$

Parallel prefix [24] realization of $X + Y$, where $X = a_{n-1} \dots a_0$, and $Y = b_{n-1} \dots b_0$, primarily derives propagate $p_i = a_i \vee b_i$ and generate $g_i = a_i b_i$ signals. These are entered into a network of logical nodes that produce group propagate P and group generate G signals, which lead to positional carries $G_{i-1:0}$ (i.e., the generated carry within positions 0 to $i-1$). The sum bits are then obtained as $s_i = h_i \oplus G_{i-1:0}$, in parallel, where $h_i = a_i \oplus b_i$. Eqn. 4 can be realized via such parallel prefix network (PPN), which is augmented by an extra row of simpler logical nodes that enforce EAC within the computation of the actual carries $c_i = G_{i-1:0} \vee P_{i-1:0} w_n$, where $w_n = G_{n-1:0}$. Fig. 1 that is reproduced from [3] depicts such RPP structure, based on Kogge-Stone (KS) PPN, where the required logical nodes are described in Fig. 2.

A similar, but slightly more complex, RPP design for modulo-$(2^n - 3)$ adders, which takes advantage of double representation of residues in $[0, 2]$, is offered in [13].

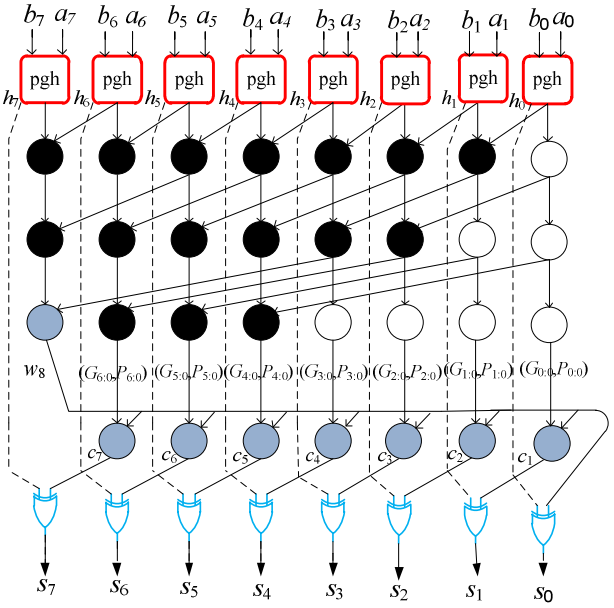

Figure 1. Modulo-$(2^n - 1)$ RPP adder

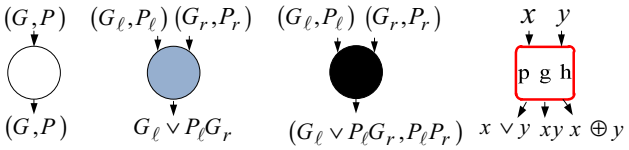

Figure 2. Basic cells used in Fig. 1

B. General modulo-$(2^n - \delta)$ adder

There are several proposals for efficient implementation of modulo-m adders in the relevant literature, where without loss of practical generality, the authors opt on moduli of the form $m = 2^n - \delta$ [4-10]. All these adders are based on Eqn. 3, above. Most of them use w_n (i.e., the most significant bit of $W = X + Y + \delta$) to select one of the two sums, which are obtained in parallel [6, 8], or in sequence [4-5].

Hiasat [7] computes two sets of propagate (p) and generate (g) signals for $X + Y$ and W. The latter set is used by a simplified carry look-ahead logic to solely compute w_n, which is used as the selector of one of the (p, g) sets. The selected set is used in a carry look-ahead architecture to lead to the desired sum.

The extreme case of $\delta = 2^{n-2} + 1$ is studied in [11]. In this work, the carry-out of W is recognized as the carry of $X + Y + 1$, or that of $X + Y + 2^{n-2}$, where both cannot be 1 simultaneously, since $W \le 2(2^n - \delta - 1) + \delta = 2^n + 2^{n-1} + 2^{n-2} - 3 < 2^{n+1}$. The PPN for positions 0, to $n-3$ is shared for the two adders that leads to significant area saving, where the carries for $X + Y$ are easily obtained from the carries of $X + Y + 1$. The sum bits corresponding to $X + Y$ and W are selected via the carry-out of the latter.

Shang Ma et al. [10] have set up a special PPN (with a preprocessing carry save stage), for the case of $\delta = 2^q + 1 (1 \le q \le n-2)$ that primarily computes the positional carry signals of W.

These carries are then corrected to lead to carries of $X + Y$, via removing the effect of the two 1-valued bits in positions 0 and q. A simple logic is employed, per binary position, which combines the two carries to obtain the required actual carry.

Finally, Jaberipur et. al [9] compute W and store $\overline{w_n}$ along $|W|_{2^n}$ that indicates whether the correct sum is $|W|_{2^n}$ ($w_n = 1$) or $W - \delta$ ($w_n = 0$). However, the required subtraction is postponed and fused with the next addition.

None of the aforementioned contributions make use of the EAC of $X + Y$ addition, as is common for the case of $\delta = 1$ [3], and recently used for $\delta = 3$[13].

III. THE PROPOSED MODULAR ADDITION SCHEME

The proposed modulo-$(2^n - \delta)$ addition scheme is based on the following Eqn. 5, for $\delta = 2^q + 1$ $(1 \le q \le n-2)$, where $W = X + Y$.

$$|X+Y|_{2^n-\delta} = \begin{cases} W, & \text{if } W < 2^n \\ |W+\delta|_{2^n}, & \text{if } W \ge 2^n \end{cases} = w_{n-1} \dots w_0 + \delta w_n \quad (5)$$

Since in cases of $2^n - \delta \le W \le 2^n - 1$, we have $|X+Y|_{2^n-\delta} = W$, the output residues in $[0, 2^q]$ can also assume excess-$(2^n - \delta)$ codes in $[2^n - \delta, 2^n - 1]$, respectively. These codes, as justified by Eqn. set 6, do not jeopardize the subsequent addition or multiplication operations, since the result of corresponding operations on these codes (i.e., $l + 2^n - \delta$, for $0 \le l < \delta$) is the same as for the normal codes.

$$\begin{aligned} |X + l + 2^n - \delta|_{2^n-\delta} &= |X + l|_{2^n-\delta}, \\ |X \times (l + 2^n - \delta)|_{2^n-\delta} &= |X \times l|_{2^n-\delta} \end{aligned} \quad (6)$$

Table I contains the bit variables that are involved in the implementation of Eqn. 5.

TABLE I. BIT ORGANIZATION OF $S = |X + Y + (2^q + 1)w_n|_{2^n}$

X	x_{n-1}	…	x_q	…	x_1	x_0
Y	y_{n-1}	…	y_q	…	y_1	y_0
$X+Y$	w_{n-1}	…	w_q	…	w_1	w_0
δ EAC			w_n			w_n
S	s_{n-1}	…	s_q	…	s_1	s_0

To avoid the two n-bit additions suggested by Table I, the q least significant bits of the final sum can be obtained via a parallel prefix q-bit addition $x_{q-1} \dots x_0 + y_{q-1} \dots y_0 + w_n$, based on the RPP architecture of Fig. 1, tailored for q bits. However, there are two carry signals entering position q, namely; w_n and the carry-out of the latter q-bit addition. The maximum collective value of these carries plus those of x_q and y_q amounts to 4. Therefore, no conventional parallel prefix architecture, for positions q to $n-1$ can handle the two carry-in signals.

To remedy this problem, we devise a partial carry-save preprocessing stage, whose output bits are shown in Table II, as $h_i = x_i \oplus y_i$, and $g_i = x_i y_i$, for $q \le i \le n-1$.

TABLE II. STAGES OF $|X + Y|_{2^n-2^q-1}$ ADDITION SCHEME

X		x_{n-1}	…	x_q	x_{q-1}	…	x_1	x_0
Y		y_{n-1}	…	y_q	y_{q-1}	…	y_1	y_0
X'		h_{n-1}	…	h_q	x_{q-1}	…	x_1	x_0
Y'	g_{n-1}	g_{n-2}	…	0	y_{q-1}	…	y_1	y_0
X'		h_{n-1}	…	h_q	x_{q-1}	…	x_1	x_0
Y''		g_{n-2}	…		y_{q-1}	…	y_1	y_0
δ EAC				e				e
		h'_{n-1}	…	h'_q	h'_{q-1}	…	h'_1	h'_0
		c_{n-1}	…	c_q	c_{q-1}	…	c_1	c_0
S		s_{n-1}	…	s_q	s_{q-1}	…	s_1	s_0

This Table further contains the bits of conceptual stages of the proposed modulo-$(2^n - 2^q - 1)$ addition $|X + Y|_{2^n-2^q-1}$, where $X' + Y' = X + Y$, the actual EAC for $|X' + Y'|_{2^n-2^q-1}$ is $e = G'_{n-1:0} \vee g_{n-1}$, since $G'_{n-1:0} g_{n-1} = 0$ due to $X' + Y' = X + Y \le 2^n + 2^n - (2^{q+1} + 4) < 2^{n+1}$. Let c_q denotes the carry into position q, of the third part of Table II, within the addition $X' + Y'' + \delta e$, where $Y'' = Y' - 2^n g_{n-1}$, and in contrast to position q of Table I, the collective value of the present bits(i.e., h_q, e, and c_q) is at most 3. This desirable property helps in devising an augmented PPN for generation of carry signals c_i that are needed to obtain the final sum bits $s_i = h'_i \oplus c_i$ $(0 \le i \le n-1)$, where $h'_i = h_i$ $(0 \le i < q)$, $h'_q = h_q \oplus e$, and $h'_i = h_i \oplus g_{i-1}$ $(q+1 \le i \le n-1)$. Eqn. 7, supported by Eqn. set 8, defines the required c_is, with due justification to follow with the support of Lemma 1.

$$c_i = \begin{cases} G_{n-1:0}, & \text{if } i = 0 \\ G_{i-1:0} \vee P_{i-1:0} G_{n-1:0}, & \text{if } 1 \le i \le q \\ h_q G_{q-1:0} \vee P''_{q:0} G_{n-1:0,} & \text{if } i = q+1 \\ h_{i-1} G_{i-2:0} \vee P''_{i-1:0} G_{n-1:0}, & \text{if } i > q+1 \end{cases} \quad (7)$$

$$\begin{aligned} P''_{q:0} &= h_q \vee P_{q-1:0} \vee G_{q-1:0} \\ P''_{i-1:0} &= P'_{i-1:q+1} P''_{q:0} \end{aligned} \quad (8)$$

Lemma 1: $(G'_{i:j+1}, P'_{i:j+1} h_j) = (h_i G_{i-1:j}, H_{i:j})$, $j \ge q$, where $(G_{i:j}\ P_{i:j})$, and $(G'_{i:j}, P'_{i:j})$ are the group (generate, propagate) signals of hypothetical PPNs for $X + Y$, and $X' + Y''$, with $Y'' = Y' - 2^n g_{n-1}$, and $H_{i:j} = \bigwedge_{k=i}^{j} h_k$.

Proof (by induction on $i - j$):
Let g' and p' refer to positional counterparts of $(G'_{i:j}, P'_{i:j})$.

Basis $(i - j = 1)$:
$G'_{j+1:j+1} = g'_{j+1} = h_{j+1} g_j = h_{j+1} G_{j:j}$.
$P'_{j+1:j+1} h_j = p'_{j+1} h_j = (h_{j+1} \vee g_j) h_j = H_{j+1:j}$.

Induction: We show that the propositions $G'_{i:j+1} = h_i G_{i-1:j}$, and $P'_{i:j+1} h_j = H_{i:j}$ imply $G'_{i+1:j+1} = h_{i+1} G_{i:j}$, and $P'_{i+1:j+1} h_j = H_{i+1:j}$, respectively, as follows.
$G'_{i+1:j+1} = g'_{i+1} \vee p'_{i+1} G'_{i:j+1} = h_{i+1} g_i \vee (h_{i+1} \vee g_i) h_i G_{i-1:j}$
$= h_{i+1}(g_i \vee h_i G_{i-1:j}) = h_{i+1} G_{i:j}$.
$P'_{i+1:j+1} h_j = p'_{i+1} P'_{i:j+1} h_j = (h_{i+1} \vee g_i) H_{i:j}$
$= h_{i+1} H_{i:j} \vee g_i h_i H_{i-1:j} = H_{i+1:j}$. ■

Justification of Eqn. 7:

$\boldsymbol{i = 0}$: Recalling $(G'_{i:j+1}, P'_{i:j+1}h_j) = (h_i G_{i-1:j}, H_{i:j})$ from Lemma 1, we have
$c_0 = e = g_{n-1} \vee G'_{n-1:0} = g_{n-1} \vee G'_{n-1:q+1} \vee P'_{n-1:q+1}G'_{q:0}$
$= g_{n-1} \vee h_{n-1}G_{n-2:q} \vee H_{n-1:q}G_{q-1:0} = g_{n-1} \vee$
$h_{n-1}(G_{n-2:q} \vee H_{n-2:q}G_{q-1:0}) = G_{n-1:0}$. That is the EAC is equal to that of the original $X + Y$.

$\boldsymbol{1 \le i \le q}$: $c_i = G_{i-1:0} \vee P_{i-1:0}G_{n-1:0}$, since the attendant bits are those of the original X and Y.

$\boldsymbol{i = q + 1}$: $c_{q+1} = (h_q \vee e)c_q \vee h_q e$
$= (h_q \vee G_{n-1:0})(G_{q-1:0} \vee P_{q-1:0}G_{n-1:0}) \vee h_q G_{n-1:0}$
$= h_q G_{q-1:0} \vee P''_{q:0}G_{n-1:0}$.

$\boldsymbol{i > q + 1}$:
$c_i = G'_{i-1:q+1} \vee P'_{i-1:q+1}c_{q+1}$
$= h_{i-1}G_{i-2:q} \vee P'_{i-1:q+1}(h_q G_{q-1:0} \vee P''_{q:0}G_{n-1:0})$
$= h_{i-1}G_{i-2:q} \vee P'_{i-1:q+1}h_q G_{q-1:0} \vee P'_{i-1:q+1}P''_{q:0}G_{n-1:0}$
$= h_{i-1}G_{i-2:q} \vee H_{i-1:q}G_{q-1:0} \vee P''_{i-1:0}G_{n-1:0}$
$= h_{i-1}(G_{i-2:q} \vee H_{i-2:q}G_{q-1:0}) \vee P''_{i-1:0}G_{n-1:0}$
$= h_{i-1}G_{i-2:0} \vee P''_{i-1:0}G_{n-1:0}$. ■

A. Proposed RPP architectures for $q \le n/2$ and $q > n/2$

In the normal RPP circuit (see Fig. 1) all carry signals are available $2\Delta G$ after the last level G signals. In the current design, the same is true for $i \le q$, as is evident by Eqn. 7. However, for $i > q$, two cases are recognized depending on the value of q:

- $\boldsymbol{q \le n/2}$: The signal $P''_{q:0}$ $(= h_q \vee P_{q-1:0} \vee G_{q-1:0})$ is delivered one ΔG sooner than the final last level G signals, since $G_{q-1:0}$ is produced in the level before last of the PPN. This leads to delivery of $c_{i>q}$ also at the desired time (see the corresponding equations for $i > q$, within Eqn. 7). Fig. 3a depicts the required circuitry, for $n = 8$, and $q = 4$, followed by the description of each incorporated logical cell, in Fig. 3b. Note that no PPN nodes are present in position 6 to lead to $G_{6:0}$, since $G_{n-2:0}$ does not occur in Eqn. 7.
- $\boldsymbol{q > n/2}$: The signal $P''_{q:0}$ is ready one ΔG later than the last level G signals. However, to achieve the same time delivery, Eqn. set 8 for $q > n/2$ and $i > q$ can be modified, as in Eqn. set 9. This is achieved via $(\lceil \log n \rceil - 1)$ especial twin nodes [25] for computing $(G_{n/2-1:0} \vee P_{n/2-1})$, and devising a mixed KS/Ladner-Fischer PPN architecture, to generate $G_{q-1:n/2}$, as in Fig. 4a (for $n = 8$, and $q = 5$), followed by legends of the utilized cells (Fig. 4b). Note that the first two PPN levels use KS architecture, while the bottom one follows that of Ladner-Fischer. Therefore, some area savings (mainly due to additional buffer nodes), which grows with n, is expected in comparison to Fig. 3.

$$P''_{q:0} = h_q \vee P_{q-1:0} \vee G_{q-1:0}$$
$$= h_q \vee P_{q-1:n/2}P_{n/2-1:0} \vee G_{q-1:n/2} \vee P_{q-1:n/2}G_{n/2-1:0}$$
$$= (h_q \vee G_{q-1:n/2}) \vee P_{q-1:n/2}(G_{n/2-1:0} \vee P_{n/2-1:0}) \quad (9)$$
$$P''_{i-1:0} = P'_{i-1:q+1}P''_{q:0} = P'_{i-1:q+1}(h_q \vee G_{q-1:n/2})$$
$$\vee P'_{i-1:q+1}P_{q-1:n/2}(G_{n/2-1:0} \vee P_{n/2-1:0})$$

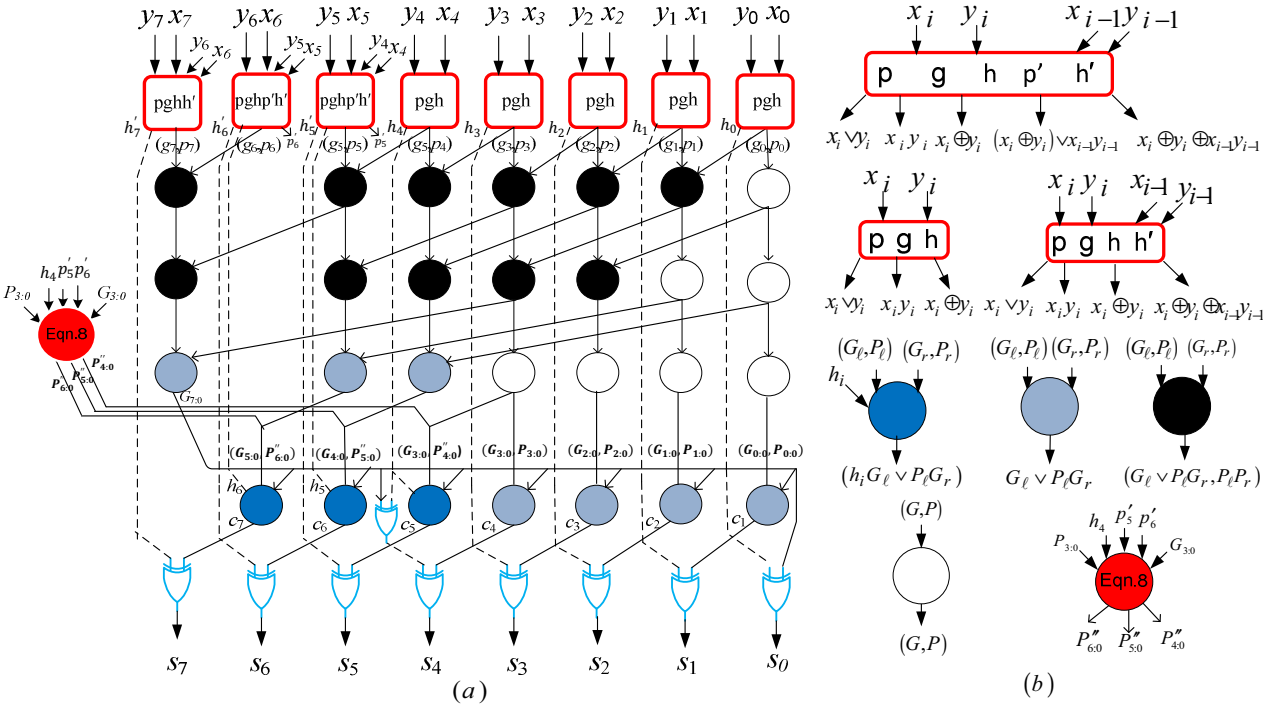

Figure 3. a: Modulo-$(2^n - 2^q - 1)$ adder for $n = 8$, $q = 4$, b: The utilized cells

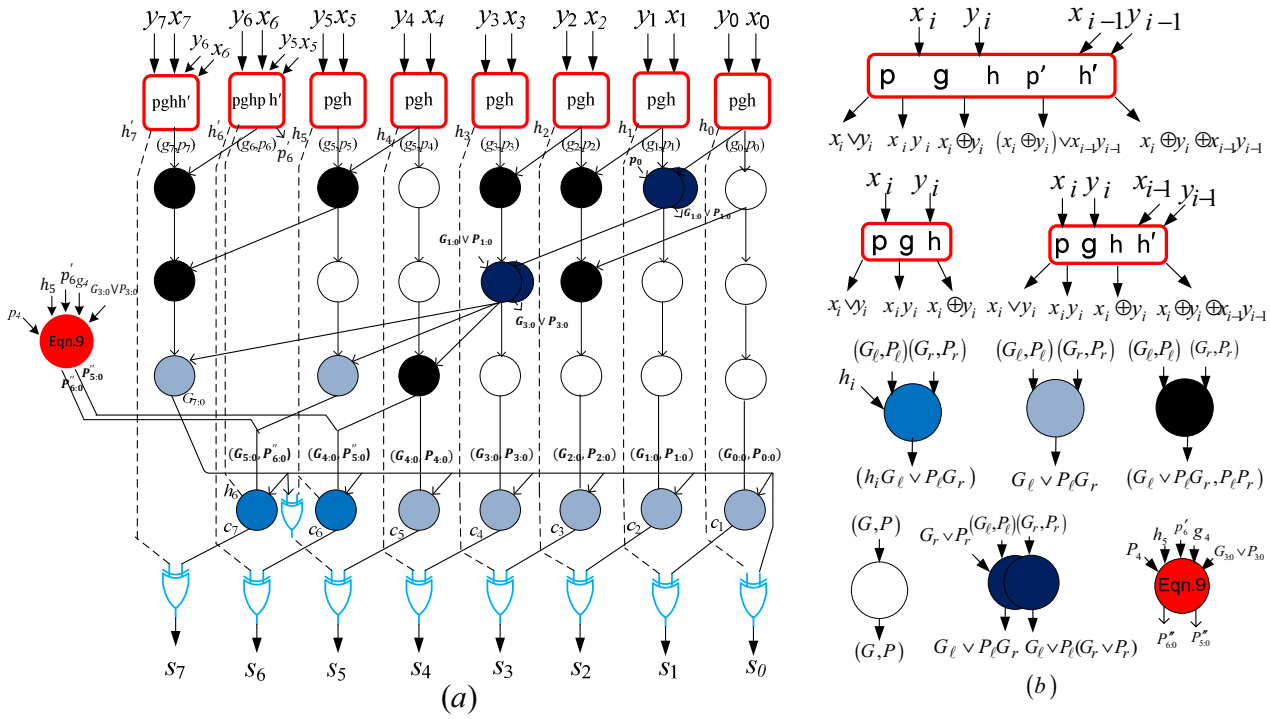

Figure 4. a: Modulo-($\mathbf{2^n - 2^q - 1}$) adder for $\boldsymbol{n = 8}$, $\boldsymbol{q = 5}$, b: The utilized cells

The c_is, in both Figs. 3 and 4, are all delivered $2\Delta G$ after the last level G signals. Therefore, all the s_i signals are available in $(5 + 2\lceil \log n \rceil)\Delta G$, with exactly the same latency as in RPP realization of modulo-$(2^n - 1)$ adders. This implies that the carry-save stage of Table II is effectively off the critical delay path.

IV. Performance Evaluation and Comparisons

The analytical gate level evaluations of several previous works and the proposed ones are reflected in Table III, where $\mathcal{A}_G$ denotes the area of a simple gate, respectively ($\mathcal{A}_{XOR} = 2$ [26]). The last entry relates to modulo-$(2^n - 1)$ RPP adder, as the one with highest performance among modulo-$(2^n - \delta)$ adders. In the works of [4-9], δ can assume any value in $[1, 2^{n-2}]$, is of the form $2^q + 1 (1 \le q \le n - 2)$ in [10], in the proposed work, and in [13], for $q = 1$, and equal to $2^{n-2} + 1$, in [11]. The term $\mathcal{A}_{P''}$, in the area formula for the proposed adders, relate to the logic that computes P'' variables. To keep P'' computations off the critical delay path, we use low cost or low delay realizations depending of value of q. In case of $q \le n/2$, we implement $P'_{i-1:q+1}$ via an AND array for $q \ge n + 1 - 2\lceil \log n \rceil$ (e.g., $q \ge 3$, for $n = 8$) that leads to $\mathcal{A}_{P''} = 2(n - q) - 2$, and by an AND tree (as in [11]), for $q < n + 1 - 2\lceil \log n \rceil$ with $\mathcal{A}_{P''} = 1.5(n - q - 2)\lceil \log(n - q - 2) \rceil + n - q$.

Likewise, for $q > n/2$, $\mathcal{A}_{P''} = 5(n - q) - 7$, in the low cost case with $q \ge n + 3 - 2\lceil \log n \rceil$, and $\mathcal{A}_{P''} = 1.5(n - q - 2)\lceil \log(n - q - 2) \rceil + 4(n - q) - 5$, for low delay case with $q < n + 3 - 2\lceil \log n \rceil$.

We have undertaken a similar evaluation for the work of [10], where in the low cost realization $\mathcal{A}_c = 2(n - q) + q - 2$, for $n + 1 - 2\lceil \log n \rceil \le q \le 2\lceil \log n \rceil$, and otherwise, $\mathcal{A}_c = 1.5(n - q - 2)\lceil \log(n - q - 2) \rceil + 1.5(q - 1)\lceil \log(q - 1) \rceil + n - q + 1$. Fig. 5 shows (for $n = 16$) that as q grows, our $\mathcal{A}_{P''}$ drops quite faster than $\mathcal{A}_c$ of [10].

TABLE III. Performance Measures of Modulo-$(2^n - \delta)$ Adders

Design	Delay (ΔG)	Area ($\mathcal{A}_G$)
[4]	$4\lceil \log n \rceil + 8$	$6n\lceil \log n \rceil + 5n + 1$
[5]	$4\lceil \log n \rceil + 12$	$3n\lceil \log n \rceil + 10n + 1$
[6]	$2\lceil \log (n - 1) \rceil + 7$	$3(n - 1)\lceil \log n \rceil + 3(n - 1)\lceil \log (n - 1) \rceil + 5n + 1$
[7]	$2\lceil \log (n - 1) \rceil + 2\lceil \log (n - 2) \rceil + 8$	$3(n - 2)\lceil \log (n - 2) \rceil - \lceil \log (n - 1) \rceil + 7n + 27$
[8]	$2\lceil \log (n - 1) \rceil + 7$	$3(n)\lceil \log (n - 1) \rceil + 13n - 5$
[9]	$2\lceil \log n \rceil + 7$	$(3n - 1)\lceil \log n \rceil + 11.5n + 1$
[10]	$2\lceil \log n \rceil + 7$	$3n\lceil \log n \rceil + 6n - 3q + 2 + \mathcal{A}_c$
[11] ($q = n - 2$)	$2\lceil \log n \rceil + 5$	$3n\lceil \log n \rceil + 1.5n\lceil \log (n - 1) \rceil + 7n + 2^{\lceil \log (n-1) \rceil - 1}$
[13]-RPP ($q = 1$)	$2\lceil \log (n - 1) \rceil + 6$	$3(n - 1)\lceil \log(n - 1) \rceil + 8n - 1$
New ($q \le n/2$)	$2\lceil \log n \rceil + 5$	$3(n - 1)\lceil \log n \rceil + 7n - 3q - 1 + \mathcal{A}_{P''}$
New ($q > n/2$)	$2\lceil \log n \rceil + 5$	$3(n - 1)\lceil \log n \rceil + 5.5n - 3q + 2\lceil \log n \rceil + \mathcal{A}_{P''}$
[3]-RPP	$2\lceil \log n \rceil + 5$	$3n\lceil \log n \rceil + 4n$

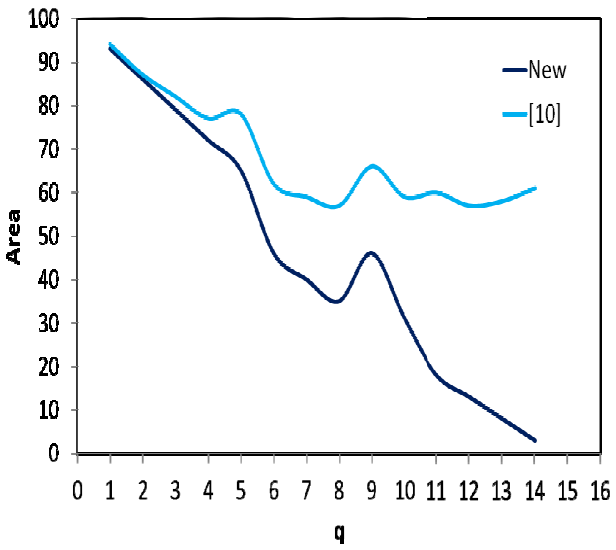

Figure 5. $\mathcal{A}_{P''}$, and $\mathcal{A}_c$ versus q

In order to confirm the above analytical evaluations, we have described the works of [6, 8-11, 13], and those of proposed ones with HDL code, and synthesized them with TSMC 0.13µm standard CMOS technology by Synopsys Design Compiler. The results are compiled as follows. Performance measures for the proposed design is shown in Table IV, for $n = 8$, and $1 \leq q \leq 6$, where area and power measures degrade, as q grows, since more buffer nodes bloom (see Fig. 4).

TABLE IV. PERFORMANCE MEASURES OF THE PROPOSED DESIGNS

q	1	2	3	4	5	6
Least delay (ns)	0.71	0.71	0.72	0.71	0.70	0.70
Area (μm^2)	1405	1268	1224	1129	1110	910
Power (μw)	449	408	383	353	326	264

Similar performance measures are compiled in Tables V-VIII, for $n = 8$, and $n = 16$, with minimum and maximum q values, where the proposed designs meet the least delay that could be achieved by the synthesis tool, with no area or power penalty with respect to other works. In particular the AT and PDP measures, for the proposed adders, are at least 21%, and 24% less than those of previous modulo-$(2^n - 2^q - 1)$ designs, respectively. Moreover, at least 60% (75%), and 10% (8%) AT (PDP) improvement is experienced with respect to the previous especial case designs for $q = n - 2$ [11], and $q = 1$ [13].

TABLE V. SYNTHESIS RESULTS FOR $(n = 8, q = 1)$

Design	**Least delay**		**Area**			**Power**		
	(ns)	Ratio	(μm^2)	Ratio	AT	(μw)	Ratio	PDP
[6]	0.77	1.08	1828	1.30	1.41	577	1.28	1.39
[8]	0.84	1.18	1501	1.07	1.26	499	1.11	1.31
[9]	0.83	1.17	1524	1.08	1.27	500	1.11	1.30
[10]	0.84	1.18	1504	1.07	1.27	500	1.11	1.32
[13]	0.80	1.12	1400	1.00	1.12	446	0.99	1.12
New	0.71	1.00	1405	1.00	1.00	449	1.00	1.00

TABLE VI. SYNTHESIS RESULTS FOR $(n = 16, q = 1)$

Design	**Least delay**		**Area**			**Power**		
	(ns)	Ratio	(μm^2)	Ratio	AT	(μw)	Ratio	PDP
[6]	0.88	1.03	4589	1.41	1.46	1383	1.40	1.44
[8]	0.90	1.06	3725	1.15	1.21	1190	1.20	1.27
[9]	0.92	1.08	3828	1.18	1.28	1295	1.31	1.41
[10]	0.97	1.14	3394	1.04	1.19	1075	1.08	1.24
[13]	0.89	1.05	3414	1.05	1.10	1018	1.03	1.08
New	0.85	1.00	3248	1.00	1.00	991	1.00	1.00

TABLE VII. SYNTHESIS RESULTS FOR $(n = 8, q = 6)$

Design	**Least delay**		**Area**			**Power**		
	(ns)	Ratio	(μm^2)	Ratio	AT	(μw)	Ratio	PDP
[6]	0.77	1.10	1825	2.00	2.21	550	2.08	2.29
[8]	0.84	1.20	1404	1.54	1.85	434	1.64	1.97
[9]	0.88	1.26	1338	1.47	1.85	454	1.72	2.16
[10]	0.75	1.07	1480	1.63	1.74	452	1.71	1.83
[11]	0.73	1.04	1394	1.53	1.60	442	1.67	1.75
New	0.70	1.00	910	1.00	1.00	264	1.00	1.00

TABLE VIII. SYNTHESIS RESULTS FOR $(n = 16, q = 14)$

Design	**Least delay**		**Area**			**Power**		
	(ns)	Ratio	(μm^2)	Ratio	AT	(μw)	Ratio	PDP
[6]	0.90	1.06	4538	2.18	2.31	1360	2.34	2.48
[8]	0.91	1.07	3988	1.91	2.05	1262	2.17	2.32
[9]	0.90	1.06	3855	1.85	1.96	1341	2.31	2.44
[10]	0.89	1.05	3394	1.63	1.71	940	1.62	1.69
[11]	0.87	1.02	3829	1.84	1.88	1226	2.11	2.16
New	0.85	1.00	2083	1.00	1.00	581	1.00	1.00

The same results are also illustrated by Figs. 6-13, where the superiority of the proposed designs is evident. In particular, Figs. 6-9, for $q = 1$, show the performance advantage of the New design with respect to the RPP adder of [13]. Moreover, extra curves corresponding to modulo-$(2^n - 1)$ RPP adder are included in order to show the minimal delay overhead of our designs, where the corresponding exact figures are compiled in Table IX.

TABLE IX. OVERHEAD WITH RESPECT TO MODULO-$(2^n - 1)$ RPP ADDER

Design	**n**	**q**	**Least delay** (ns)	Ratio	**Area** (μm^2)	Ratio	AT	**Power** (µw)	Ratio	PDP
New	8	1	0.71	1.11	1405	1.22	1.35	449	1.38	1.53
		6	0.70	1.09	910	0.79	0.86	264	0.81	0.89
[3]-RPP		-	0.64	1.00	1154	1.00	1.00	326	1.00	1.00
New	16	1	0.85	1.02	3248	1.42	1.46	991	1.68	1.72
		14	0.85	1.02	2083	0.91	0.93	581	0.99	1.01
[3]-RPP		-	0.83	1.00	2282	1.00	1.00	589	1.00	1.00

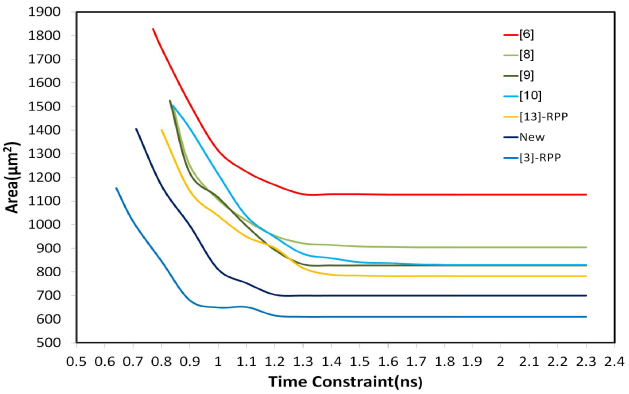

Figure 6. Area comparisons for $n = 8, q = 1$

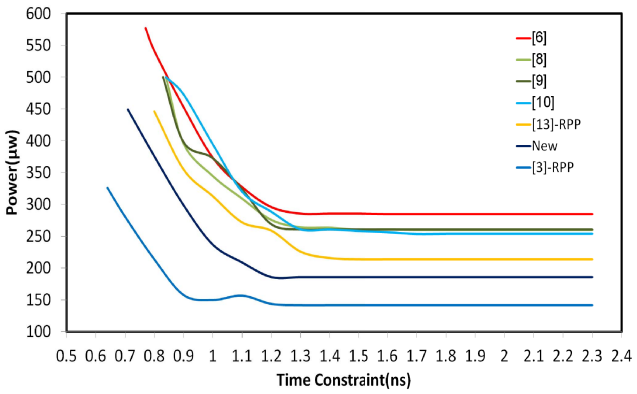

Figure 7. Power comparisons for $n = 8, q = 1$

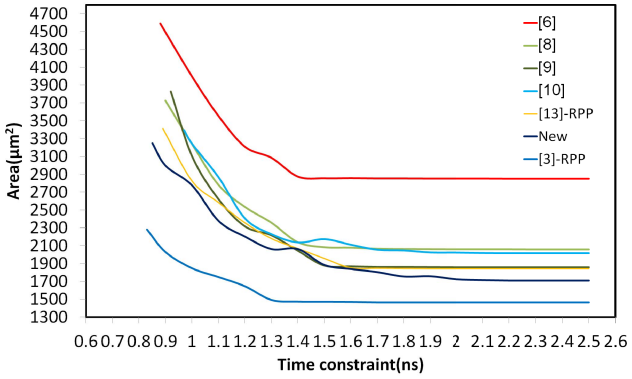

Figure 8. Area comparisons for $n = 16, q = 1$

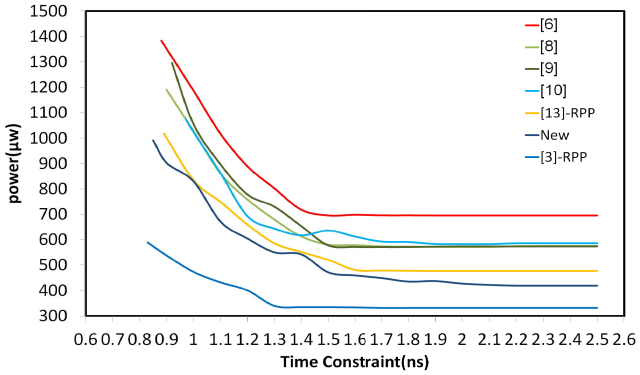

Figure 9. Power comparisons for $n = 16, q = 1$

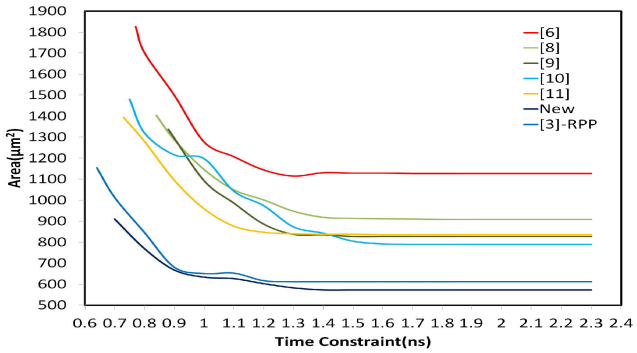

Figure 10. Area comparisons for $n = 8, q = 6$

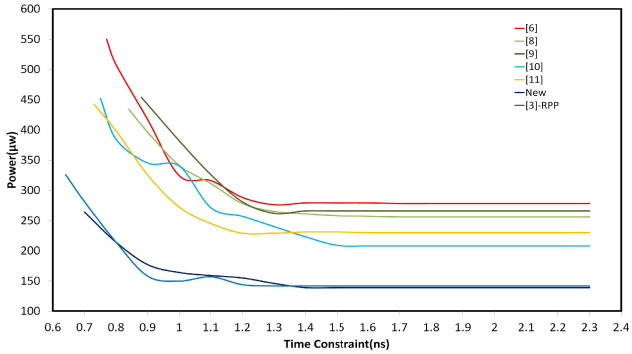

Figure 11. Power comparisons for $n = 8, q = 6$

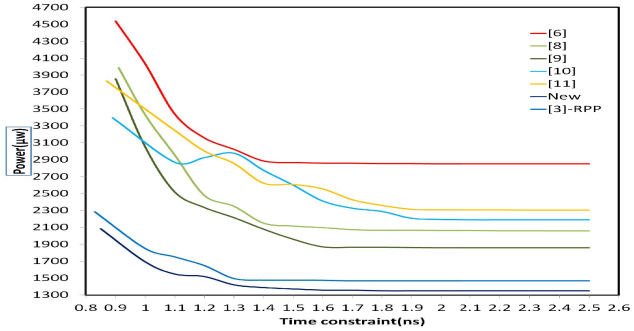

Figure 12. Area comparisons for $n = 16, q = 14$

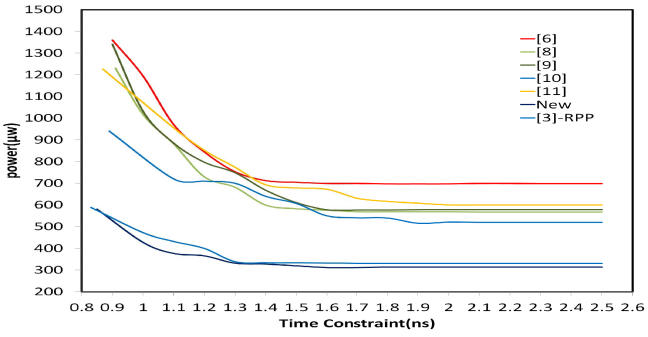

Figure 13. Power comparisons for $n = 16, q = 14$

V. CONCLUSION

Efficient realization of modulo-$(2^n - \delta)$ arithmetic operations can help in devising fast RNS arithmetic systems with several computational channels with balanced latency. For example, the moduli set $\{31, 29, 27, 25, 23, 19, 17\}$, for $n = 5$, and δ in $\{1, 3, 5, 7, 9, 13, 15\}$, respectively. Parallel prefix realization of modulo-$(2^n - \delta)$ adders, for odd $\delta > 1$, confront the difficulty of handling multiple 2^i-weighted reentrant carries. Previous solutions for $|X + Y|_{2^n - \delta}$ rely on partial or full computations towards obtaining two interim sums $X + Y$, and $X + Y + \delta$. Furthermore, some recent designs work for restricted values of δ such as $\delta = 2^{n-2} + 1$ [11], and $\delta = 2^q + 1$ [10] (e.g., $\delta = 3, 5, 9$, for $n = 5$). However, following the case of $q = 1$ in [13], via excess-$(2^q + 1)$ representation of residues in $[0, 2^q]$ (as well as normal representation), we design the required modular adders based on only one interim sum (i.e., $X + Y$). This leads to significant performance improvement on all figures of merit. Latency of the proposed adder is $(5 + 2\lceil \log n \rceil)\Delta G$, which is equal to that of similar realization (i.e., RPP) for modulo-$(2^n - 1)$ adders, and less than that of similar special case of $q = 1$ in [13]. The latency of fastest previous general modulo-$(2^n - 2^q - 1)$ adder [10] is $(7 + 2\lceil \log n \rceil)\Delta G$, while it consumes considerable additional area and dissipates more power. For example, in case of $n = 16$, and $q = 14$, it is 5% slower, consumes 63% more area, and dissipates 62% more power. Other cases can be checked in Tables V-VIII that show superiority of the proposed designs.

Also the adder of [11], for the single case of $q = n - 2$, experiences the same delay as ours, but at the cost of 84% (111%) more area (power) consumption. Furthermore, the proposed adder (for the highest value of q) is more area- and power-efficient (see Table IX) than the RPP realization of modulo-$(2^n - 1)$ adder [3], but with slight additional delay penalty (e.g., 9% area and 1% power savings at the cost of 2% more delay, for $n = 16$, and $q = 14$).

As for the future relevant research, modulo-$(2^n - 2^q - 1)$ multipliers and the corresponding totally parallel prefix (TPP in the terminology of [3]) adders can be studied.

REFERENCES

[1] D. Schinianakis, T. Stouraitis, "Multifunction Residue Architectures for Cryptography," *IEEE Trans. Circuits Syst. I, Reg. papers*, vol. 61, no. 4, pp. 1156-1169, Apr. 2014.

[2] R. Chokshi, K. S. Berezowski, A. Shrivastava, S. J. Piestrak, "Exploiting residue number system for power-efficient digital signal processing in embedded processors," in *Proc. of the international conference on Compilers, architecture, and synthesis for embedded systems* (*CASES*), pp. 19-28, Oct. 2009.

[3] L. Kalampoukas, D. Nikolos, C. Efstathiou, H. T. Vergos, J. Kalamatianos, "High-Speed Parallel-Prefix Modulo $2^n - 1$ Adders," *IEEE Trans. Comput.* , vol. 49, no. 7, pp. 673-680, July 2000.

[4] M. A. Bayoumi, G. A. Jullien, W. C. Miller, "A VLSI implementation of residue adders," *IEEE Trans. Circuits Syst.* , vol. 34 , no. 3, pp. 284 -288, Mar. 1987.

[5] M. Dugdale, "VLSI implementation of residue adders based on binary adders," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.* , vol. 39, no. 5, pp. 325-329, May 1992.

[6] S. J. Piestrak, "Design of high-speed residue-to-binary number system converter based on Chinese remainder theorem," in *Proc. of the Internatnal Conference on Computer Design*, pp. 508-511, Oct. 1994.

[7] A. A. Hiasat, "High-speed and reduced-area modular adder structures for RNS," *IEEE Trans. Comput.* , vol. 51, no. 1, pp. 84 -89, Jan. 2002.

[8] R. A. Patel, M. Benaissa, N. Powell, S. Boussakta, "Novel Power-Delay-Area-Efficient Approach to Generic Modular Addition," *IEEE Trans. Circuits Syst. I, Reg. papers*, vol. 54, No. 6, pp. 1279-1292, June 2007.

[9] G. Jaberipur, B. Parhami, S. Nejati, "On building general modular adders from standard binary arithmetic components," in *Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers,* pp. 154-159, Nov. 2011.

[10] S. Ma, J. H. Hu, C. H. Wang, "A Novel Modulo-($2^n - 2^k - 1$) Adder for Residue Number System," *IEEE Trans. Circuits Syst. I, Reg. papers*, vol. 60, no. 11, pp. 2962-2972, Nov. 2013.

[11] R. A. Patel, M. Benaissa, S. Boussakta, "Fast Modulo $2^n - (2^{n-2} + 1)$ Addition: A New Class of Adder for RNS," *IEEE Trans. Comput.* , vol. 56, no. 4, pp. 572-576, April 2007.

[12] H. Fatemi, G. Jaberipur, "Double representation Modulo-$(2^n - 3)$ adders," in *Proc. of the 21st International Conference on Systems, Signals, and Image Processing*, pp. 119-122, May 2014.

[13] G. Jaberipur, S. H. F. Langroudi, "$(4 + 2\log n)$ Delta-G Parallel Prefix Modulo-$(2^n - 3)$ Adder via Double Representation of Residues in [0,2]," *IEEE Trans. Circuits Syst. II, Exp. Briefs* , vol. 62, doi: 10.1109/TCSII.2015.2407772.

[14] 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, Aug. 1994.

[15] S. J. Piestrak, "Design of multi-residue generators using shared logic," in *Proc. of the IEEE International Symposium on Circuits and Systems (ISCAS)* , pp. 1435-1438, May 2011.

[16] J. Y. S. Low, Chip-Hong Chang, "A New Approach to the Design of Efficient Residue Generators for Arbitrary Moduli," *IEEE Trans. Circuits Syst. I, Reg. papers*, vol. 60, no. 9, pp. 2366-2374, Feb. 2013.

[17] A. A. Hiasat, "RNS arithmetic multiplier for medium and large moduli," *IEEE Trans.Circuits Syst. II Analog and Digit. Signal Process.* , vol. 47, no. 9, pp. 937-940, Sept. 2000.

[18] A. A. Hiasat, "A Suggestion for a Fast Residue Multiplier for a Family of Moduli of the Form $(2^n - 2^p \pm 1)$," *The Computer J.* , vol. 47, no. 1,pp. 93-102, 2004.

[19] L. Li, J. Hu, Y. Chen, "An universal architecture for designing modulo $2^n - 2^k - 1$ multiplier," *IEICE Electron. Expr.* , vol. 9, no. 3, pp. 193-199, Feb. 2012.

[20] J. C. Bajard, M. Kaihara, T. Plantard, "Selected RNS Bases for Modular Multiplication," in *Proc. Of the 19th IEEE Symposium on Computer Arithmetic(ARITH),* pp. 25-32, June 2009.

[21] A. Nannarelli, M. Re, G. C. Cardarilli, "Tradeoffs between residue number system and traditional FIR filters," in *Proc. of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 305-308, May 2001.

[22] P. Patronik, K. Berezowski, S. J. Piestrak, J. Biernat, A. Shrivastava, "Fast and energy-efficient constant-coefficient FIR filters using residue number system," in *Proc. of the International Symposium on Low Power Electronics and Design,* pp. 385-390, Aug. 2011.

[23] G. C. Cardarilli, A. Nannarelli, M. Re, "Residue Number System for Low-Power DSP Applications," in *Conference Record of the Forty-First Asilomar Conference on Signals, Systems and Computers (ACSSC),* pp.1412-1416, Nov. 2007.

[24] D. Harris, "A taxonomy of parallel prefix networks," in *Conference Record of the Thirty-Seventh Asilomar Conference on Signals, Systems and Computers,* pp. 2213-2217, Nov. 2003.

[25] R. A. Patel, M. Benaissa, S. Boussakta, "Fast Parallel-Prefix Architectures for Modulo $2^n - 1$ Addition with a Single Representation of Zero," *IEEE Trans. Comput.*, vol. 56, no. 11, pp. 1484-1492, 2007.

[26] R. Zimmermann, "Efficient VLSI implementation of modulo $2^n \pm 1$ addition and multiplication," in *Proc. Symposium on Computer Arithmetic*, pp. 158-167, April 1999.