The CSI Journal on
Computer Science and Engineering
Vol. 13, No. 2, 2016
Pages 48-53
Regular Paper

The Impact of Excess-Modulo Representation of Residues on Modulo-$(2^n - 5)$ Parallel Prefix Addition

Ghassem Jaberipur Hassan Ghasemi Motlagh

Department of Computer Science and Engineering, Shahid Beheshti University, Tehran, Iran

Abstract

It is desirable to realize latency-balanced computation channels in residue number systems (RNS). In particular modulo-$(2^n - \delta)$ addition with selected values of δ (e.g., $\delta \in \{1, 3, 5\}$) is of interest. Modulo-$(2^n - 1)$ adders are realized via one's complement addition, where $(2^n - 1)$ is also valid as a second representation for zero. Fast parallel prefix realization of such adders exist with $(3 + 2\lceil \log n \rceil)$ ΔG latency, where ΔG denotes the delay of a simple 2-input gate. Similarly, modulo-$(2^n - 3)$ adders with excess-modulo representations of residues in $\{0, 1, 2\}$ has been recently proposed with $(4 + 2\lceil \log n \rceil)$ ΔG latency. It has been shown that such double representation of residues does not jeopardize the subsequent RNS operations. To approach larger dynamic ranges, while keeping the channel widths (i.e., n) as low as possible, fast modulo-$(2^n - 5)$ adders could be quite useful. That's how we are motivated to explore the design and implementation of such adders with the goal of achieving the same latency of $(4 + 2\lceil \log n \rceil)$ ΔG. The proposed scheme is basically the same as that of the aforementioned modulo-$(2^n - 3)$ adders. However, there are particular non-trivial challenges to be overcome. The proposed scheme is analyzed for latency and area measures, which are confirmed via circuit synthesis by Synopsis Design Compiler.

Keywords: Modulo-$(2^n - 5)$ Adder, Excess-Modulo Representation, Parallel Prefix Modular Adders, Digital Signal Processing.

1. Introduction

The latency balance within the parallel computation channels of a residue number system (RNS) arithmetic architecture is desirable [1]. On the other hand, limited dynamic range of the classical moduli set $\tau = \{2^n - 1, 2^n, 2^n + 1\}$ calls for additional balanced moduli, where modular addition and multiplication can be performed with the same speed as in the original three moduli. Parallel prefix fast adders [2-5] for the aforementioned τ system with $(3 + 2\lceil \log n \rceil)$ ΔG latency has been proposed [6], where ΔG refers to the delay of a simple 2-input AND or OR gate. The corresponding modulo-$(2^n - 1)$ adder takes advantage of a second representation for zero, as $2^n - 1$, where a regular parallel prefix (RPP) and the so called totally parallel prefix (TPP) architectures [7] lead to $(5 + 2\lceil \log n \rceil)$ and $(3 + 2\lceil \log n \rceil)$ ΔG latencies, respectively.

Moreover, similar modulo-$(2^n - 3)$ RPP and TPP adders with $(6 + 2\lceil \log n \rceil)$ and $(4 + 2\lceil \log n \rceil)$ ΔG latencies have also been reported [8], where 0, 1 and 2 have alternative representations, as $2^n - 3$, $2^n - 2$, and $2^n - 1$, respectively. However, we have not encountered any special modulo-$(2^n - 5)$ adder design in the literature, thus motivated to present one with the same performance as that of the aforementioned modulo-$(2^n - 3)$ adder. The rest of this paper deals with a background on RNS arithmetic in Section 2, also with a brief description of previous relevant modular adders, offers our proposed modulo- $(2^n - 5)$ adder designs in Section 3, evaluations and comparisons in Section 4, and finally concluding remarks in Section 5.

2. Background

A residue number system is mainly defined by a k-moduli set $\{m_1, \ldots m_k\}$, where the modulus are commonly pairwise prime to allow for maximal range (aka dynamic range) of presentable numbers [9]. The dynamic range, as such, is

equal to $M = \prod_{i=1}^{k} m_i$. The RNS representation of a binary number X is composed of k residues $x_i = |X|_{m_i}$ (i.e., the remainder of integer division $\left\lfloor \frac{X}{m_i} \right\rfloor$, for $1 \le i \le k$), where addition and multiplication on binary operands X and Y are performed on their corresponding residues x_i and y_i in k parallel data paths.

Modulo-2^n addition is exactly the same as conventional n-bit addition, while multiplication is even simpler and less costly. However, given the mutually prime property, only one m_i can be a power of two. The next popular moduli are of the form $2^{n\pm p} - 1$ ($n \gg p$), since the required adder is exactly the same as a one's complement adder, and the main component of the corresponding multiplier is an $(n \pm p)$-operand modular adder. The $n \gg p$ restriction is to keep the moduli set balanced in terms of arithmetic speed. However, the number of such mutually prime moduli are limited. Therefore, moduli of the form $2^{n\pm p} + 1$, and more recently modulo-$(2^n - 3)$ have been employed in order to set up high dynamic range balanced moduli sets [8].

Parallel prefix modular addition is popular, since it provides for more balanced arithmetic circuits. For example, $(3 + 2\lceil \log n \rceil)$ ΔG TPP adders have been proposed for modulo-$(2^n - 1)$, where figure 1 (borrowed from [8]) depicts the required architecture. Also the fastest modulo-$(2^n - 3)$ adder that we have encountered is due to [8], where the latency of its TPP architecture (see figure 2 that is borrowed from [8]) amounts to $(4 + 2\lceil \log n \rceil)$ ΔG. The legend for these figures and the proposed designs are compiled in table 1.

3. The Proposed Modulo-$(2^n - 5)$ Adder

The modulo-$(2^n - 1)$, and $-(2^n - 3)$ adders are fairly balanced, since for instance in case of n = 8 (i.e., a typical bit length in RNS applications such as image processing [10]), 9, and 10 ΔG latencies are achieved, respectively. Nevertheless, in this section, we propose an RPP modulo-$(2^n - 5)$ adder, with the same $(6 + 2\lceil \log n \rceil)$ latency as the fastest previous modulo-$(2^n - 3)$ RPP adder [8], which consumes slightly more area and dissipates marginally more power. We also provide a TPP version of the proposed adder that performs as fast as the previous modulo-$(2^n - 3)$ TPP adder [8], while the additional area and power measures are negligible.

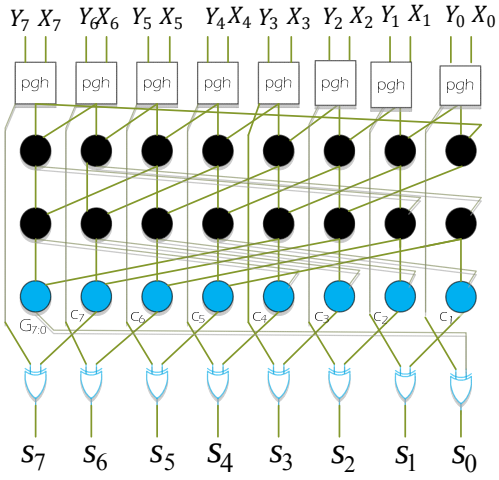

Figure 1. The TPP modulo- $(2^n - 1)$ adderarchitecture

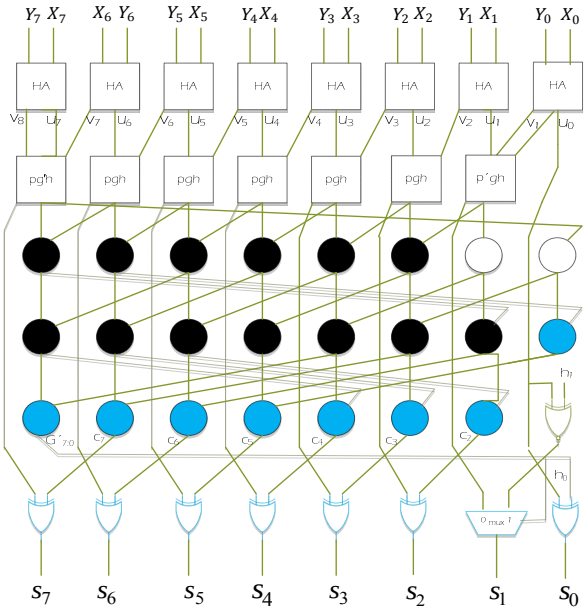

Figure 2. The TPP modulo-$(2^n - 3)$ adder architecture

Table 1. Logical cells used in the respected figures

Legend	Description	Figures
(G, P)  (G, P)	Buffer node	2, 3
$(G_\ell, P_\ell)\ (G_r, P_r)$  $G_\ell \vee P_\ell G_r$	G node	1, 2, 3, 4
$(G_\ell, P_\ell)\ (G_r, P_r)$  $(G_\ell \vee P_\ell G_r, P_\ell P_r)$	(G, P) node	1, 2, 3, 4
$(G_l, P_l)\ (G_r, P_r)$ P_{r-1}  $(G_l v P_l G_r, P_l v G_r v P_r P_{r-1})$	(G, P′) node	3, 4
$x\ \ y$ HA $xy\ \ x \oplus y$	Half adder	2, 3, 4
$v\ \ u$ pgh $p = u \vee v$ $g = u \wedge v$ $h = u \oplus v$	p = u ∨ v g = u ∧ v h = p$\bar{g}$	1, 2, 3, 4
$u_n\ \ u_{n-1} v_{n-1}$ pg'h $u_{n-1} \vee v_{n-1}$ $u_{n-1} \oplus v_{n-1}$ $(u_{n-1} \wedge v_{n-1}) \vee v_n$	$g' = g \vee v_n$	2, 3, 4
$u_1\ \ v_1\ u_0$ P'gh $u_0 \vee v_1 \vee u_1$ $u_1 \oplus v_1$ $u_1 v_1$	$p' = p \vee u_0$	2

3.1. The Proposed Modulo-$(2^n - 5)$ RPP Adder

We follow the same approach as in [8] and come up with table 2 that describes the steps of modulo-$(2^n - 5)$ addition $S = |X + Y|_{2^n-5}$, where EAC stands for end-around carry. This table is quite similar to the figure 4 of [8], except that position 2 (instead of 1) is occupied by $G'_{n-1:0} = v_n + G_{n-1:0}$, where $G_{n-1:0}$ denotes the generated carry-out of $U + V'$ [11]. Such difference, however, leads to different equations for carry signals c_1 to c_3 with respect to the corresponding ones in [8]. These equations are derived as Eqns. 1-3, where $c_3 \leq 1$ is the generated carry for $u_2 + v_2 + G'_{n-1:0} + c_2 \leq 3$, since v_2 and c_2 cannot be 1 at once. The reason is that$c_2 = u_1v_1 + (u_1 + v_1)c_1 = u_1(v_1u_0G'_{n-1:0}) + v_1u_0G'_{n-1:0}$, since $v_2u_1 = 0$ and $v_1u_0 = 0$. Therefore, c_3 is in fact the carry of summation $u_2 + (v_2 \vee c_2) + G'_{n-1:0}$. Note that Eqn. 3 includes the modified group propagate signal $P'_{2:0} = p_2 \vee G_{1:0} \vee P_{1:0}$. Also, the fact that $g_0 = 0$, and $p_0 = u_0$, leads to $G_{2:0} = G_{2:1}$, and simpler equation for $P'_{2:0}$ as in Eqn. 4.

Figure 3 depicts an RPP realization of such carry signals and the corresponding sum signals for $n = 8$, where the preprocessing stage consists of an array of half adders, and thus leads to $(7 + 2\lceil \log n \rceil)\,\Delta G$ latency. Note that derivation of $s_2 = h_2 \oplus G'_{n-1:0} \oplus c_2$ requires an extra XOR operation, which does not lengthen the critical delay path (CDP). The rest of carry equations are described by Eqn. 5, where the group propagation signals $P_{i-1:0}$ (for $i > 3$) are based on $P'_{2:0}$ (i.e., $P_{i-1:0} = P_{i-1:3}P'_{2:0}$).

Table 2. Modulo-$(2^n - 5)$ EAC addition with carry-save preprocessing

X Y		x_{n-1} ···· x_2 x_1 x_0 y_{n-1} ···· y_2 y_1 y_0			
U V		u_{n-1} ···· u_2 u_1 u_0 v_n v_{n-1} ···· v_2 v_1 0			
V′ 5EAC		u_{n-1} ···· u_2 u_1 u_0 v_{n-1} ···· v_2 v_1 0 $G'_{n-1;0}$ $G'_{n-1;0}$			
		h_{n-1} ···· h_2 h_1 h_0 c_{n-1} ···· c_2 c_1 c_0			
S		s_{n-1} ···· s_2 s_1 s_0			

$$c_1 = u_0G'_{n-1:0} \tag{1}$$

$$c_2 = G_{1:0} \vee P_{1:0}G'_{n-1:0} \tag{2}$$

$$\begin{aligned} c_3 &= u_2v_2 \vee u_2c_2 \vee (u_2 \vee v_2 \vee c_2)G'_{n-1:0} \\ &= u_2v_2 \vee (u_2 \vee v_2)G'_{n-1:0} \vee (u_2 \vee v_2 \vee G'_{n-1:0})c_2 \\ &= g_2 \vee p_2G'_{n-1:0} \vee p_2(G_{1:0} \vee P_{1:0}G'_{n-1:0}) \vee c_2G'_{n-1:0} \\ &= g_2 \vee p_2G_{1:0} \vee (p_2 \vee c_2)G'_{n-1:0} \\ &= G_{2:0} \vee (p_2 \vee G_{1:0} \vee P_{1:0}G'_{n-1:0})G'_{n-1:0} \\ &= G_{2:0} \vee (p_2 \vee G_{1:0} \vee P_{1:0})G'_{n-1:0} \\ &= G_{2:0} \vee P'_{2:0}G'_{n-1:0} \end{aligned} \tag{3}$$

$$\begin{aligned} P'_{2:0} &= p_2 \vee g_1 \vee p_1p_0 = p_2 \vee u_1v_1 \vee (u_1 \vee v_1)u_0 \\ &= p_2 \vee u_1(v_1 \vee u_0) = p_2 \vee u_1(x_0y_0 \vee (x_0 \vee y_0)\overline{x_0y_0}) \\ &= p_2 \vee u_1(x_0 \vee y_0) \end{aligned} \tag{4}$$

$$c_i = G_{i-1:0} \vee P_{i-1:0}G'_{n-1:0} \tag{5}$$

1) Further Speedup of RPP Architecture

Recalling the $(6 + 2\lceil \log n \rceil)\,\Delta G$ latency of the modulo-$(2^n - 3)$ adder of [8], we can use the technique, therein, to achieve the same $1\Delta G$ gain. Eqn. 6 is reproduced from [8], where $u'_{i-1} = x_{i-1} \vee y_{i-1}$, $p'_i = u_i$, and $g'_{i-1} = u'_{i-1}v_{i-1}$. Therefore, $\Delta G_{i:i-1}$ is reduced to $4\Delta G$, since p'_i and g'_{i-1} are delivered in $2\Delta G$, and thus the total delay for the proposed RPP adder is also $(6 + 2\lceil \log n \rceil)\,\Delta G$.

$$G_{i:i-1} = g_i \vee p_ig_{i-1} = g_i \vee p'_ig'_{i-1} \tag{6}$$

3.2. Corresponding TPP Architecture

The TPP architectures are expected to lead to $2\,\Delta G$ less overall delay with respect to the corresponding RPP architectures, as is the case in the TPP modulo-$(2^n - 3)$ adder of [8]. Therefore, we elaborate here on developing a TPP modulo-$(2^n - 5)$ adder that yields all the sum bits in $(4 + 2\lceil \log n \rceil)\,\Delta G$, where we use Eqn. 7 that describes the general TPP carry formula, with the understanding that $P_{i-1:0} = P_{i-1:3}P'_{2:0}$. As such the required TPP equations for $n = 8$ are described below.

$$\begin{aligned} c_i &= G_{i-1:0} \vee P_{i-1:0}G'_{n-1:i} \\ c_1 &= g_0 \vee p_0G'_{7:1} \\ &= g_0 \vee p_0\left(g'_7 \vee p_7(G_{6:5} \vee p_{6:5}(G_{4:3} \vee p_{4:3}G_{2:1}))\right) \\ c_2 &= G_{1:0} \vee p'_{1:0}G'_{7:2} \\ &= G_{1:0} \vee p'_{1:0}(G'_{7:6} \vee p_{7:6}(G_{5:4} \vee p_{5:4}G_{3:2})) \\ c_3 &= G_{2:0} \vee p'_{2:0}G'_{7:3} \\ &= G_{2:1} \vee p'_{2:1}\left(g_0 \vee p_0(g'_7 \vee p_7(G_{6:5} \vee p_{6:5}G_{4:3}))\right) \\ c_4 &= G_{3:0} \vee p_{3:0}G'_{7:4} \\ &= G_{3:2} \vee p'_{3:2}(G_{1:0} \vee p'_{1:0}(G'_{7:6} \vee p_{7:6}G_{5:4})) \\ c_5 &= G_{4:0} \vee p_{4:0}G'_{7:5} \\ &= G_{4:3} \vee p_{4:3}\left(G_{2:1} \vee p'_{2:1}(g_0 \vee p_0(g'_7 \vee p_7G_{6:5}))\right) \\ c_6 &= G_{5:0} \vee p_{5:0}G'_{7:6} \\ &= G_{5:4} \vee p_{5:4}\left(G_{3:2} \vee p'_{3:2}(G_{1:0} \vee p'_{1:0}G'_{7:6})\right) \\ c_7 &= G_{6:0} \vee p_{6:0}g'_7 \\ &= G_{6:5} \vee p_{6:5}\left(G_{4:3} \vee p_{4:3}(G_{2:1} \vee p'_{2:1}(g_0 \vee p_0g'_7))\right) \end{aligned} \tag{7}$$

The corresponding circuitry is depicted by figure 4, where the delay figures are indicated along the data paths. In particular, s_1 and s_5 are delivered in $11\Delta G$ and s_2 in $12\Delta G$, while the other sum signals are available at $10\Delta G$. The reason for the extra delays for s_1 and s_5 is $1\Delta G$ delay of g'_7 with respect to other g_i signals that leads to $5\Delta G$ delay of $G'_{0:7}$, and that of s_2 is due to the extra XOR. The former can be fixed via implementing Eqn. set 8 and the latter by Eqn. 9. Regarding s_2, one of the two possible sum values fo $G'_{n-1:0} =$ 0r or 1 (i.e., $h_2 \oplus G_{1:0}$ or $h_2 \oplus \overline{G_{1:0} + P_{1:0}}$) is selected by a multiplexer, and thus $\Delta s_2 = 2\Delta G + \Delta G'_{n-1:0}$, as the other s_i outputs (see figure 5). As for g'_7, the modified equation (i.e., Eqn. set 8) delivers it in $3\Delta G$, and thu $G'_{0:7}$ s is available in $4\Delta G$, as other $G_{i:i-1}$ signals.

$$\begin{aligned} g'_7 &= g_7 \vee v_8 = u_7v_7 \vee v_8 = (x_7 \oplus y_7)v_7 \vee x_7y_7 \\ &= (x_7 \vee y_7)v_7 \vee x_7y_7 = v_8 \vee u'_7v_8 \\ G'_{0:7} &= g_0 \vee p_0g'_7 \end{aligned} \tag{8}$$

$$\begin{aligned} s_2 &= h_2 \oplus G'_{n-1:0} \oplus c_2 \\ &= h_2 \oplus G'_{n-1:0} \oplus (G_{1:0} \vee P_{1:0}G'_{n-1:0}) \\ &= h_2 \oplus (G'_{n-1:0}\overline{G_{1:0} + P_{1:0}} + G_{1:0}\overline{G'_{n-1:0}}) \end{aligned}$$

$$= (h_2 \oplus \overline{G_{1:0} + P_{1:0}})G'_{n-1:0} + (h_2 \oplus G_{1:0})\overline{G'_{n-1:0}} \quad (9)$$

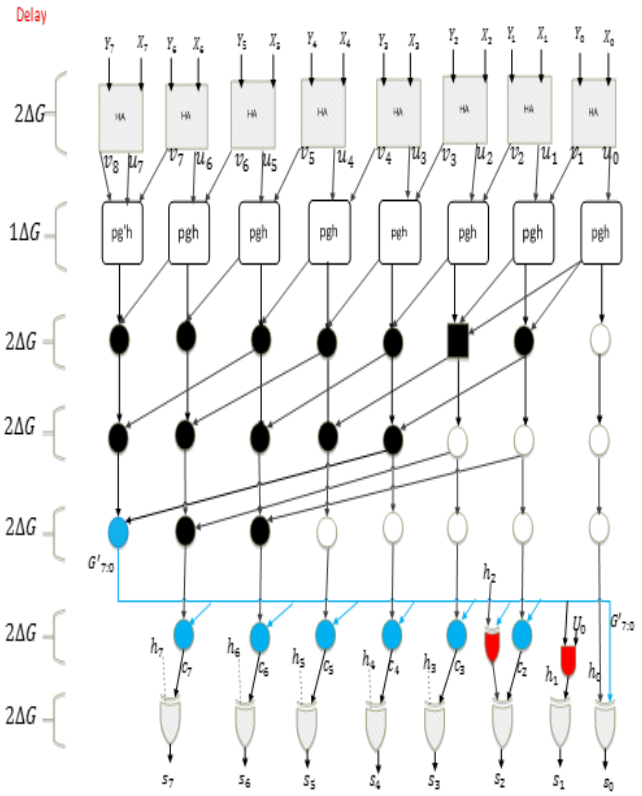

Figure 3. The proposed RPP modulo-$(2^n - 5)$ adder

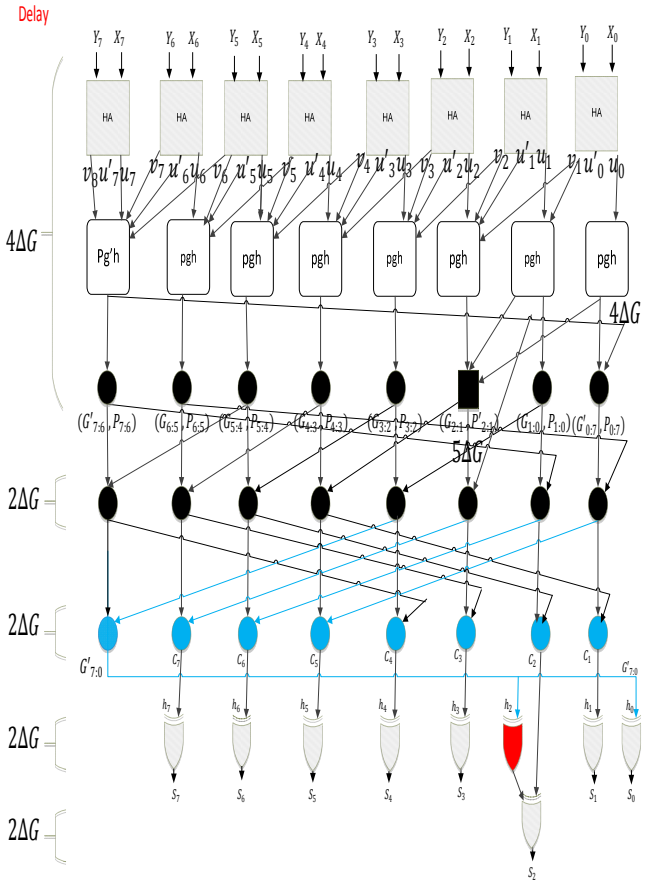

Figure 4. The proposed TPP modulo-$(2^n - 5)$ adder

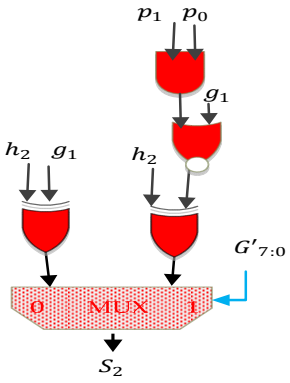

Figure 5. Modification for the terminal part of the s_2 path in figure 4

3.3. Addition of Two Excess-Modulo Residues

Most of RNS applications require modular addition and multiplications, where at least one operand is furnished via input data, which is therefore normally represented (i.e., single representation of residues). In such cases, Eqn. set 10 is meant to show that excess-modulo representation of the other operand does not jeopardize the resulted residue, where $A \in [0, 2^n - 5]$, $B \in [2^n - 5, 2^n - 1]$, and $B' = B - (2^n - 5)$, represent the normal representation of B. However, the same analysis fails for the case of both operands in excess-modulo representations, which can be handled as follows, in hypothetical cases of occurrence.

$$|A + B|_{2^n-5} = |A + B' + 2^n - 5|_{2^n-5} = |A + B'|_{2^n-5},$$
$$|A \times B|_{2^n-5} = |A \times (B' + 2^n - 5)|_{2^n-5} = |A \times B'|_{2^n-5} \quad (10)$$

The excess-modulo operand pairs $2^n - \delta_1$ and $2^n - \delta_2$, where $1 \le \delta_1, \delta_2 \le 5$, should be examined for possible wrong sums. Let $A + B = G_{n-1:0}w_{n-1} \dots w_0$, and $S = s_n s_{n-1} \dots s_0 = w_{n-1} \dots w_0 + 5G_{n-1:0}$. Therefore, $s_n = w_{n-1}c_{n-1} = (h_{n-1} \oplus G_{n-2:0})c_{n-1}$, where all the required three operands are available in the RPP and TPP realizations of figure 3 and 4. On the other hand, $s_{n-1} \dots s_0 = |A + B|_{2^n-5} = |2^n - \delta_1 + 2^n - \delta_2|_{2^n-5}$ denotes the sum that is produced by the proposed adders, where the EAC $= G'_{n-1:0} = 1$. Therefore, Eqn. set 11 holds, where the cases of $S \ge 2^n$ (i.e., $S \in \{2^n, 2^n + 1, 2^n + 2, 2^n + 3\}$), denoted by $s_n = 1$, are not valid, and should be corrected by another subtraction by $2^n - 5$, or actually addition by 5. However, this correction act affects only the four least significant bits of the sum (i.e., $s_3 s_2 s_1 s_0$), as is justified by the content of table 3.

$$S = 2^n - \delta_1 + 2^n - \delta_2 - (2^n - 5), 2^n - 5 \le S \le 2^n + 3 \quad (11)$$

Table 3. Two excess-$(2^n - 5)$ operands

A, B	$s_3 s_2 s_1 s_0$	$s'_3 s'_2 s'_1 s'_0$
$2^n - 4, 2^n - 1$	0000	0101
$2^n - 3, 2^n - 2$	0000	0101
$2^n - 3, 2^n - 1$	0001	0110
$2^n - 2, 2^n - 2$	0001	0110
$2^n - 2, 2^n - 1$	0010	0111
$2^n - 1, 2^n - 1$	0011	1000

Eqn. set 12 describes the corrected sum bits $s'_i (0 \le i \le 3)$ in terms of the original sum bits s_i. Note that $s_n = 0$ in other

cases that are not listed in table 3. Therefore, additional latency of the correction logic is 4ΔG.

$$s'_0 = s_0 \oplus s_n, s'_1 = s_1\overline{s_n} \vee (s_1 \oplus s_0)s_n, s'_2 = s_2\overline{s_n} \vee \overline{s_1 s_0}s_n, s'_3 = s_3\overline{s_n} \vee s_1 s_0 s_n \quad (12)$$

4. Evaluations and Comparisons

Recalling the modifications to figure 4 that was explained in Section 3.2, the overall latency of the proposed TPP modulo-$(2^n - 5)$ adder is equal to the desired $(4 + 2\lceil \log n \rceil)\Delta G$ in the practical cases, where at most one of the operands is excess-modulo. Also that of the RPP realization of figure 3 is $(6 + 2\lceil \log n \rceil)\Delta G$. Therefore, the proposed adders are latency balanced with the corresponding previous modulo-$(2^n - 3)$ adders of [8]. The gate level delay and area measures of the reference and proposed adders are listed in table 4. To confirm the results, therein, we have coded function of the four adders, perform correction tests, and simulated the corresponding circuits by TSMC .9 μm Synopsys Design Compiler. Figures 6-8 depict the area and power curves in terms of time constraints of the synthesis tool, for $n \in \{8, 16, 32\}$, respectively. These results confirm the latency balance between modulo-$(2^n - 3)$ and -$(2^n - 5)$ channels, where the area consumption and power dissipation are also compatible. The corresponding measures for the least met time constraint are compiled in tables 5-7.

Table 4. Analytical gate-level performance measures

Design	Delay (ΔG)	Area (ΔA)
RPP $(2^n - 5)$	$6 + 2\lceil \log n \rceil$	$\mathbf{3n\lceil \log n \rceil + 7n + 2}$
TPP $(2^n - 5)$	$4 + 2\lceil \log n \rceil$	$\mathbf{3n\lceil \log n \rceil + 7n + 7}$
RPP $(2^n - 3)$	$6 + 2\lceil \log n \rceil$	$\mathbf{3n\lceil \log n \rceil + 7n - 6}$
TPP $(2^n - 3)$	$4 + 2\lceil \log n \rceil$	$\mathbf{3n\lceil \log n \rceil + 8n - 10}$

Table 5. Synthesis results for n = 8

Design	Delay (ns)	Area (μm²)	Power (μw)
RPP $(2^n - 5)$	0.5897	19695.16	650.87
TPP $(2^n - 5)$	0.5499	19150.68	616.61
RPP $(2^n - 3)$	0.5499	17220.08	585.71
TPP $(2^n - 3)$	0.5498	20376.62	624.14

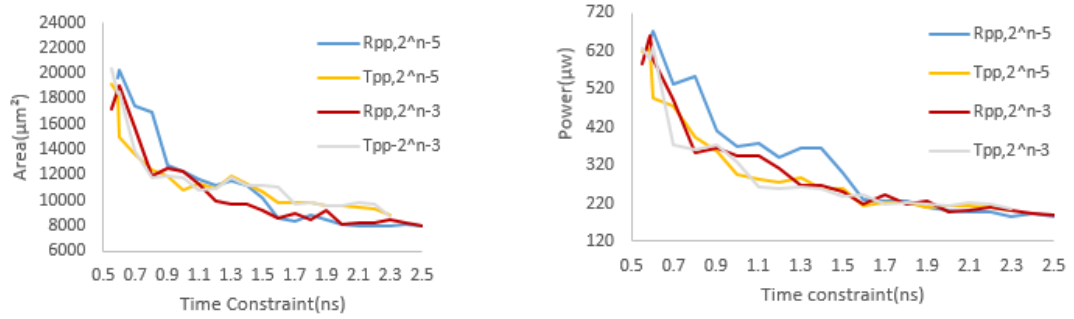

Figure 6. Area and power comparisons for n = 8

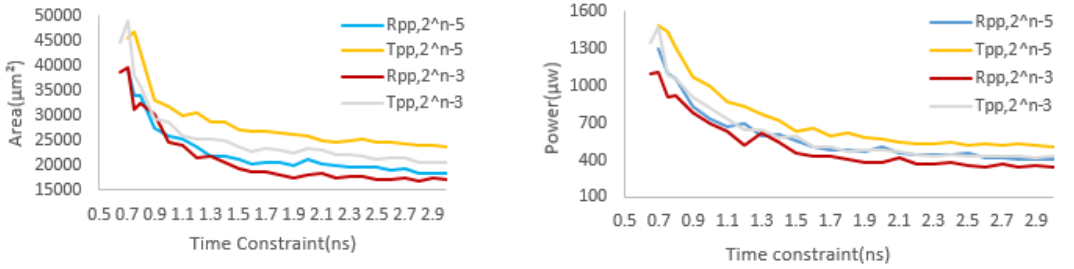

Figure 7. Area and power comparisons for n = 16

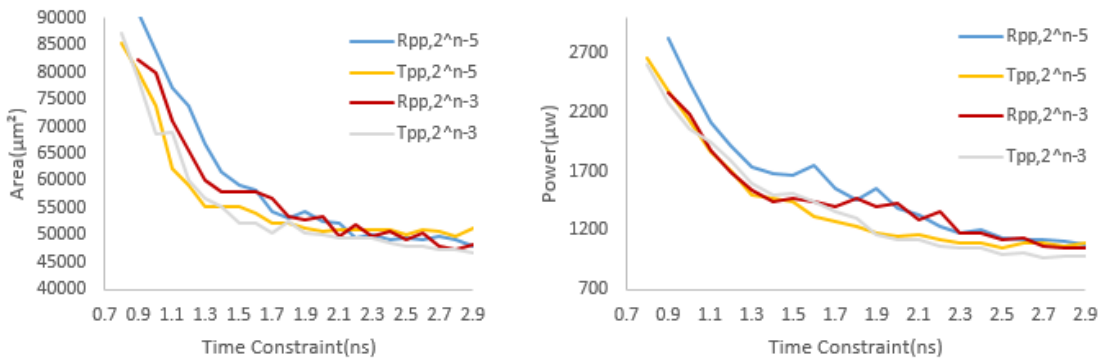

Figure 8. Area and power comparisons for n = 32

Table 6. Synthesis results for n = 16

Design	Delay (ns)	Area (μm^2)	Power (µw)
RPP (2^n-5)	0.7000	39286.79	1301
TPP (2^n-5)	0.6999	45391.73	1486.8
RPP (2^n-3)	0.6500	38553.01	1098.4
TPP (2^n-3)	0.6499	44467.15	1350.8

Table 7. Synthesis results for n = 32

Design	Delay (ns)	Area (μm^2)	Power (µw)
RPP (2^n-5)	0.89	91145.98	2822.4
TPP (2^n-5)	0.85	84412.00	2460.2
RPP (2^n-3)	0.90	82295.13	2363.9
TPP (2^n-3)	0.79	87038.07	2602.3

5. Conclusions

RNS applications can be realized to show better figures of merit (especially performance) in case of balanced moduli set, where the latency of all the computation channels are almost equal (i.e., at most $1\Delta G$ difference). Modular addition is in particular of interest, where parallel prefix $(3+2\lceil\log n\rceil)\,\Delta G$ adders have been presented for moduli set $\{2^n-1, 2^n, 2^n+1\}$ [2], and also modulo-(2^n-3) adders with $(4+2\lceil\log n\rceil)\,\Delta G$ performance [8]. Additional moduli of the form $(2^n-\delta)$ are desirable to accommodate larger dynamic ranges, provided that $(4+2\lceil\log n\rceil)\,\Delta G$ adders are feasible. We presented a modulo-(2^n-5) parallel prefix adder that meets the latter performance constraint. This was shown via gate-level analysis that was confirmed by synthesis results. Such performance was achieved at marginal penalty in area and power measures.

As for the relevant future work, for extending the dynamic range while keeping the whole moduli set balanced, we plan to study the design and implementation of compatible modulo-(2^n+3) and -(2^n+5) adders.

References

[1] H. Ahmadifar, and G. Jaberipur, "A New Residue Number System with 5-Moduli Set: $\{2^{2q}, 2^q\pm3, 2^q\pm1\}$," *The Computer Journal*, to appear.

[2] R. P. Brent, and H. T. Kung, "A regular layout for parallel adders," IEEE Trans. On Computers, vol. C-31, no. 3, pp. 260-264, Mar. 1982.

[3] P. M. Kogge, and H. S. Stone, "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations," in IEEE Transactions on Computers, vol. C-22, no. 8, pp. 786-793, Aug. 1973.

[4] S. Knowles, "A family of adders," *Proceedings 14th IEEE Symposium on Computer Arithmetic*, pp. 277-281, June 2001.

[5] R. Ladner, and M. Fischer, "Parallel prefix Computation," *J. ACM*, vol. 27, no. 4, pp.831.838, Oct. 1980.

[6] G. Jaberipur, and S. Nejati, "Balanced Minimal Latency RNS Addition for Moduli Set $\{2^n-1, 2^n, 2^n+1\}$," 18th International Conference on Systems Signals and Image Processing (IWSSIP), pp. 1-7, 16-18 June 2011.

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

[8] G. Jaberipur, and 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]," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 62, no. 6, pp. 583-587, June 2015.

[9] B. Parhami, *Computer Arithmetic Algorithms and Hardware Designs*, Oxford Univ. Press, 2000.

[10] R. Chokshi, K. S. Berezowski, A. Shrivastava, and 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.

[11] C. Efstathiou, H. T. Vergos, and D. Nikolos, "Fast Parallel-Prefix Modulo 2^n+1 Adder," *IEEE Trans. Comput.*, vol. 53, no. 9, pp. 1211-1216, Sept 2004.

Ghassem Jaberipur is an Associate Professor of Computer Engineering in the Department of Computer Science and Engineering of Shahid Beheshti University, Tehran, Iran. He received his BS in electrical engineering and PhD in computer engineering from Sharif University of Technology in 1974 and 2004, respectively, (where he is recognized as one of the 50 distinguished graduates for years 1966-2016), MS in engineering from UCLA in 1976, and MS in computer science from University of Wisconsin, Madison, in 1979. His main research interest is in computer arithmetic. Dr. Jaberipur is also affiliated with the School of Computer Science, Institute for Research in Fundamental Sciences (IPM), in Tehran, Iran.
E-mail: jaberipur@sbu.ac.ir

Hassan Ghasemi Motlagh received his B.Sc. degree in Computer Engineering from Mazandaran University of Science and Technology, Mazandaran, Iran, in 2014. Now, he is working on his M.Sc. in Computer Architecture at Shahid Beheshti University. His research interests include Computer Arithmetic, RNS.
E-mail: h.ghasemimotlagh@sbu.ac.ir

Paper Handling Data:

Submitted: 07.11.2016
Received in revised form: 09.12.2016
Accepted: 28.12.2016
Corresponding author: Dr. Ghassem Jaberipur,
Department of Computer Science and Engineering,
Shahid Beheshti University, Tehran, Iran.