A Sign Detector for a Group of Three-Moduli Sets

Ahmad Hiasat

Abstract—This paper presents a structure for a sign detector that is suitable for a group of three-moduli sets. This group is defined by $(2^{n+p}, 2^n-1, 2^n+1)$, where n and p are any positive integers such that $0 \leq p < n$. The dynamic range of this group consists of $(3n+p)$ bits and the proposed structure can be easily changed to accommodate different values of p. When compared with other previously presented structures, the sign detector proposed in this paper proved to be more efficient in terms of area, delay and power.

Index Terms—Residue number system, sign detection, computer arithmetic, binary number system

1 Introduction

Residue Number System (RNS) is a number representation that enjoys the capability of processing each digit independently of other residue digits within the same moduli set. This feature allows fast arithmetic operations like addition, subtraction and multiplication [1], [2], [3]. On the other hand, RNS has few major limitations that make its advantages limited to applications which require the above mentioned arithmetic operations. The limitations include: sign detection, comparison and division. These limitations have prevented RNS from being adopted in general purpose computers. Nevertheless, RNS has been used extensively in specific Digital Signal Processing applications [1], [2], [3].

In this paper, Section 2 presents a literature review of the published work related to sign detection in RNS. Section 3 introduces sign detection basics and the approach adopted in this paper. Section 4 presents the proposed sign detection algorithm, while Section 5 presents the proposed hardware implementation of the algorithm. Section 6 performs hardware synthesis and modeling for the proposed work and for the most competitive published structures and compares the results.

The notational convention adopted in this paper is introduced and defined as follows:

- $\{m_1, m_2, \ldots\ldots, m_N\}$, moduli set of N pairwise relatively prime positive integers.
- $M = \prod_{i=1}^{N} m_i$, dynamic range.
- For any integer $X \in [0, M)$, residue representation of X is: $X \overset{RNS}{\rightarrow} (R_1, R_2, \ldots\ldots, R_N)$
- $R_i = \langle X \rangle_{m_i}$, the least non-negative remainder when dividing X by m_i.
- For the three-moduli set $(2^{n+p}, 2^n-1, 2^n+1)$, where $0 \leq p < n$, the residue digits, expressed in binary notation, are:
  - $R_1 = \sum_{i=0}^{n+p-1} r_{1i}2^i = r_{1(n+p-1)} \cdots r_{11} r_{10}$
  - $R_2 = \sum_{i=0}^{n-1} r_{2i}2^i = r_{2(n-1)} \cdots r_{21} r_{20}$
  - $R_3 = \sum_{i=0}^{n} r_{3i}2^i = r_{3n} \cdots\cdots r_{31} r_{30}$
- $\hat{m}_i = \frac{M}{m_i}$
- $\langle \frac{1}{\hat{m}_i^{-1}} \rangle_{m_i}$, multiplicative inverse of $\hat{m}_i$ (i.e., $\langle \langle \hat{m}_i \rangle_{m_i} \frac{1}{\hat{m}_i^{-1}} \rangle_{m_i} = 1$)
- $\lfloor . \rfloor$ is the largest integer less than or equal to (.).
- Logic operations: NOT, OR, AND and XOR are represented by the symbols $\overline{(.)}$, $\vee$, $\wedge$ and $\oplus$, respectively.

2 Literature Review

The general idea behind sign detection in RNS is based mainly on reverse decoding of residue digits into their binary equivalent. The binary value is then checked if it belongs to the positive or negative range. However, such an approach is very demanding [2], [3], [4]. Nevertheless, there have been considerable efforts to reduce the time and hardware requirements of sign detection operation [5], [6], [7], [8], [9], [10], [11], [12], [13], [14].

Vu [5] presented a sign detector based on fractional representation where each residue digit is applied to a ROM. The outputs of the ROMs, expressed as fractional values, are applied to multi-operand adders. The sign is the most significant bit of the sum. Alia and Martinelli [6] presented a sign detection structure that uses a base extension and requires two multi-operand modulo adders and two modulo multipliers. Tomczak [7] presented a sign detector for the moduli set $(2^n-1, 2^n, 2^n+1)$. The structure of this sign detector consists of a multi-operand adder, two carry-generation circuits and a post-processing circuit. Wang et al. [8] presented a residue to binary converter for the moduli set $(2^n-1, 2^n, 2^n+1)$ which can be easily customized to be a sign detector. It requires a 2n bit carry-save adder network and a 2n bit carry-generation circuit. Xu et al. [9] introduced an algorithm for sign detection for the moduli set $(2^n-1, 2^n, 2^n+1)$. The introduced sign detector

• *The author is with the School of Engineering, Princess Sumaya University for Technology, Amman, Jordan. E-mail: a.hiasat@psut.edu.jo.*

Manuscript received 7 Sept. 2015; revised 21 Feb. 2016; accepted 15 Mar. 2016. Date of publication 27 Mar. 2016; date of current version 14 Nov. 2016. Recommended for acceptance by C. K. Koc.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2016.2547381

requires a multi-operand adder, a carry-generation circuit, a comparator and a post-processing circuit. Sousa and Martins [10] introduced a sign detection algorithm for integers represented in the RNS extended 3-moduli set $(2^n-1, 2^{n+p}, 2^n+1)$, which is the same moduli set considered in this paper. The sign detector in [10] uses, basically, one modulo (2^n-1) adder, one modulo 2^{n+k} subtracter and a sign generation module. The work in [11] suggests a magnitude comparator, which can function as a sign detector if a residue value X is compared with $\frac{M}{2}$. However, the hardware and time requirements are considerably high as shown in [10]. Xu et al. [12] also presented the most recent work on sign detection for the moduli set $(2^{n+1}-1, 2^n-1, 2^n)$. This sign detector requires, mainly, a carry-save adder, a comparator and a carry-generation unit. Al-Radadi and Siy [13] presented a generic sign detection algorithm based on the new Chinese Remainder Theorem. When the algorithm is customized for the moduli set $(2^{n+1}-1, 2^n-1, 2^n)$, it requires a modulo adder, a carry-save adder network and carry-generation adder. Using Mixed Radix Conversion, Akkal and Siy [14] also introduced a sign detection algorithm that can deal with any moduli set. When customized to the previous moduli set, the detector requires a modulo adder, two carry-save network and carry-generation adder.

Lately, the moduli set of the form $(2^{n+p}, 2^n-1, 2^n+1)$ has received additional attention [15]. The authors in [15] have presented a new approach for mapping the moduli set into another one of the form $(2^{n+p-\ell}, 2^{n-\ell}-1, 2^{n-\ell}+1)$. The purpose of this transformation is to reduce the word length of each residue digit by ℓ bits and consequently the dynamic range, where ℓ is an integer that belongs to the interval $[1, n-2]$.

3 Sign Detection Basics and Approach

Since $X \in [0, M)$, then X is considered positive if it belongs to the range $[0, \frac{M}{2})$, and negative if it belongs to the range $[\frac{M}{2}, M)$, [1]. Recalling that $M = (2^{2n}-1)2^{n+p}$, then we can define the Sign function of X to be $S(X)$, where $S(X) = 0$ if X is positive and 1 otherwise. Hence:

$$S(X) = \begin{cases} 0, & \text{if } X \in [0, 2^{3n+p-1} - 2^{n+p-1}) \\ 1, & \text{if } X \in [2^{3n+p-1} - 2^{n+p-1}, 2^{3n+p} - 2^{n+p}). \end{cases} \quad (1)$$

In a recent publication [12], a new theorem for sign detection algorithm, that uses basically the Chinese Remainder Theorem, has been introduced for the general moduli set of the form $\{m_1, m_2, \ldots\ldots, m_N = 2^k\}$. The theorem concludes that a k bit sign indicator variable α is given by:

$$\alpha = \left\langle \sum_{i=1}^{N} \frac{m_N \langle \frac{1}{\hat{m}_i^{-1}} \rangle_{m_i}}{m_i} R_i - \frac{1}{\prod_{i=1}^{N-1} m_i} R_1 \right\rangle_{2^k}. \quad (2)$$

Based on derivations in [12] and upon subsisting $k = n+p$, (1) can be written as:

$$S(X) = \begin{cases} 0, & \text{if } \alpha \in [0, 2^{n+p-1}) \\ 1, & \text{if } \alpha \in [2^{n+p-1}, 2^{n+p}). \end{cases} \quad (3)$$

Introducing the binary representation of $\alpha = \alpha_{n+p-1} \cdots \alpha_1 \alpha_0$, and using the Most Significant Bit (MSB) of α as the sign indicator bit, then (3) can be rewritten as:

$$S(X) = \alpha_{n+p-1}. \quad (4)$$

The multiplicative inverses for the popular three-moduli set $(2^n, 2^n-1, 2^n+1)$ were introduced in [16]. It was shown that: $\langle \frac{1}{\hat{m}_1^{-1}} \rangle_{m_1} = \langle -1 \rangle_{m_1}$, $\langle \frac{1}{\hat{m}_2^{-1}} \rangle_{m_2} = \langle 2^{n-1} \rangle_{m_2}$ and $\langle \frac{1}{\hat{m}_3^{-1}} \rangle_{m_3} = \langle -2^{n-1} \rangle_{m_3}$ are the multiplicative inverses for $\hat{m}_1$, $\hat{m}_2$ and $\hat{m}_3$, respectively. It can be easily verified that changing the modulus 2^n to 2^{n+p} will cause a similar shift only in multiplicative inverses of the second and third moduli, if $n > p \geq 0$. That is, the multiplicative inverses for $(2^{n+p}, 2^n-1, 2^n+1)$, are: $\langle \frac{1}{\hat{m}_1^{-1}} \rangle_{m_1} = \langle -1 \rangle_{m_1}$, $\langle \frac{1}{\hat{m}_2^{-1}} \rangle_{m_2} = \langle 2^{n-p-1} \rangle_{m_2}$, and $\langle \frac{1}{\hat{m}_3^{-1}} \rangle_{m_3} = \langle -2^{n-p-1} \rangle_{m_3}$.

In this paper we will be using the sign detection theorem introduced in [12], along with these multiplicative inverses to develop a generalized sign detection structure for this group of moduli.

4 Proposed Sign Detector Algorithm for the Moduli Set $(2^{n+p}, 2^n-1, 2^n+1)$, General Case

The moduli set under consideration is $(2^{n+p}, 2^n-1, 2^n+1)$, and the corresponding residue digits are (R_1, R_2, R_3). Rearranging the moduli to be in the form $(m_1, \ldots, m_N = 2^k)$, then our moduli set and the corresponding residue digits will have the forms $(2^n+1, 2^n-1, 2^{n+p})$ and (R_3, R_2, R_1), respectively. Upon substituting the above listed multiplicative inverses, (2) produces:

$$\begin{aligned} \alpha = \Bigg\langle \Bigg\lfloor & \frac{2^{n+p} \langle -2^{n-p-1} \rangle_{(2^n+1)}}{(2^n+1)} R_3 \\ & + \frac{2^{n+p} \langle 2^{n-p-1} \rangle_{(2^n-1)}}{(2^n-1)} R_2 \\ & + \frac{2^{n+p} \langle -1 \rangle_{(2^{n+p})}}{(2^{n+p})} R_1 - \frac{1}{(2^{2n}-1)} R_3 \Bigg\rfloor \Bigg\rangle_{2^{n+p}}. \end{aligned} \quad (5)$$

The terms on the Right Hand Side (RHS) of (5) are rational numbers that, in their current form, are very expensive to realize in terms of hardware requirements. The intention in the following mathematical formulation is to simplify these terms and re-express them as simple integer quantities that can be added using binary adders. As will be shown, the value of α given in (5) will ultimately turn into modulus 2^{n+p} of the sum of three $(n+p)$ bits integer variables and three 1-bit values.

Simplifying (5) starts with observing that $\langle -2^{n-p-1} \rangle_{(2^n+1)} = (2^n + 1 - 2^{n-p-1})$, hence, (5) can be written as:

$$\begin{aligned} \alpha = \Big\langle \Big\lfloor & 2^{n+p} R_3 - \tfrac{2^{2n-1}}{(2^n+1)} R_3 + \tfrac{2^{2n-1}}{(2^n-1)} R_2 \\ & - R_1 - \tfrac{1}{(2^{2n}-1)} R_3 \Big\rfloor \Big\rangle_{2^{n+p}}. \end{aligned} \quad (6)$$

Since $\langle 2^{n+p}R_3\rangle_{2^{n+p}}=0$, then (6) is reduced to:

$$\alpha=\left\langle\left\lfloor-\frac{(2^{2n-1})}{(2^n+1)}R_3+\frac{2^{2n-1}}{(2^n-1)}R_2 - R_1-\frac{1}{(2^{2n}-1)}R_3\right\rfloor\right\rangle_{2^{n+p}}. \tag{7}$$

Unifying the denominator in (7) for the fractional terms to become $(2^{2n}-1)$, then it can be put in the following form:

$$\alpha=\left\langle\left\lfloor-\frac{(2^{2n-1})(2^n-1)}{(2^{2n}-1)}R_3+\frac{(2^{2n-1})(2^n+1)}{(2^{2n}-1)}R_2 -R_1-\frac{1}{(2^{2n}-1)}R_3\right\rfloor\right\rangle_{2^{n+p}}. \tag{8}$$

To further simplify (8), the first term on the RHS of (8) can be written as: $\frac{2^{2n-1}(2^n-1)R_3}{(2^{2n}-1)}=\frac{2^{2n}(2^n-1)}{(2^{2n}-1)}\frac{R_3}{2}$. Similarly, the second term on the RHS of (8) can also be written as: $\frac{2^{2n-1}(2^n+1)R_2}{(2^{2n}-1)}=\frac{2^{2n}(2^n+1)}{(2^{2n}-1)}\frac{R_2}{2}$. Such forms of these two terms will enable the use of the identity $\frac{2^{2n}}{2^{2n}-1}=1+\frac{1}{2^{2n}-1}$ later on, which helps in splitting a rational number into a quotient and a remainder.

Expressing $\frac{R_2}{2}=\lfloor\frac{R_2}{2}\rfloor+\frac{r_{20}}{2}$, and $\frac{R_3}{2}=\lfloor\frac{R_3}{2}\rfloor+\frac{r_{30}}{2}$, where r_{20} and r_{30} are the Least Significant Bits (LSB) of R_2 and R_3, respectively, and re-arranging terms, then (8) can be rewritten as:

$$\alpha=\left\langle\left\lfloor-R_1+\frac{2^{2n}(2^n+1)}{(2^{2n}-1)}\left(\left\lfloor\frac{R_2}{2}\right\rfloor+\frac{r_{20}}{2}\right) -\frac{2^{2n}(2^n-1)}{(2^{2n}-1)}\left(\left\lfloor\frac{R_3}{2}\right\rfloor+\frac{r_{30}}{2}\right) -\frac{2}{(2^{2n}-1)}\left(\left\lfloor\frac{R_3}{2}\right\rfloor+\frac{r_{30}}{2}\right)\right\rfloor\right\rangle_{2^{n+p}}. \tag{9}$$

Using the above mentioned identity (i. e., $\frac{2^{2n}}{2^{2n}-1}=1+\frac{1}{2^{2n}-1}$), then (9) can be expressed as:

$$\alpha=\langle\lfloor-R_1 +\left(1+\frac{1}{2^{2n}-1}\right)(2^n+1)\left(\left\lfloor\frac{R_2}{2}\right\rfloor+\frac{r_{20}}{2}\right) -\left(1+\frac{1}{2^{2n}-1}\right)(2^n-1)\left(\left\lfloor\frac{R_3}{2}\right\rfloor+\frac{r_{30}}{2}\right) -\frac{2}{(2^{2n}-1)}\left(\left\lfloor\frac{R_3}{2}\right\rfloor+\frac{r_{30}}{2}\right)\right\rfloor\rangle_{2^{n+p.}} \tag{10}$$

Expanding the terms on RHS of (10) results in:

$$\alpha=\left\langle\left\lfloor-R_1+(2^n+1)\left\lfloor\frac{R_2}{2}\right\rfloor-(2^n-1)\left\lfloor\frac{R_3}{2}\right\rfloor +(1+\frac{1}{2^{2n}-1})(2^{n-1}+\frac{1}{2})r_{20} -(1+\frac{1}{2^{2n}-1})(2^{n-1}-\frac{1}{2})r_{30} +\frac{(2^n+1)\left\lfloor\frac{R_2}{2}\right\rfloor-(2^n-1)\left\lfloor\frac{R_3}{2}\right\rfloor}{2^{2n}-1} -\frac{2\left(\left\lfloor\frac{R_3}{2}\right\rfloor+\frac{r_{30}}{2}\right)}{2^{2n}-1}\right\rfloor\right\rangle_{2^{n+p.}} \tag{11}$$

In order to collect all the fractional quantities in (11) in one term, we define L to be:

$$L=\lfloor L_1+L_2\rfloor, \tag{12}$$

where:

$$L_1=\left(1+\frac{1}{2^{2n}-1}\right)\left(2^{n-1}+\frac{1}{2}\right)r_{20} -\left(1+\frac{1}{2^{2n}-1}\right)\left(2^{n-1}-\frac{1}{2}\right)r_{30} \tag{13}$$

$$L_2=\frac{(2^n+1)\left\lfloor\frac{R_2}{2}\right\rfloor-(2^n-1)\left\lfloor\frac{R_3}{2}\right\rfloor}{2^{2n}-1} -\frac{2\left(\left\lfloor\frac{R_3}{2}\right\rfloor+\frac{r_{30}}{2}\right)}{2^{2n}-1}$$

$$=\frac{(2^n+1)\left\lfloor\frac{R_2}{2}\right\rfloor-(2^n+1)\left\lfloor\frac{R_3}{2}\right\rfloor}{2^{2n}-1}-\frac{r_{30}}{2^{2n}-1}$$

$$=\frac{\left\lfloor\frac{R_2}{2}\right\rfloor-\left\lfloor\frac{R_3}{2}\right\rfloor}{2^n-1}-\frac{r_{30}}{2^{2n}-1}. \tag{14}$$

By defining L, it can be observed that the floor value $\lfloor.\rfloor$ operator in (11) can now be removed. Dealing with the floor value is now restricted to computing L. Hence, applying modulus 2^{n+p} to (11) produces four integer variables in the following form:

$$\alpha=\left\langle\langle-R_1\rangle_{2^{n+p}}+\left\langle(2^n+1)\left\lfloor\frac{R_2}{2}\right\rfloor\right\rangle_{2^{n+p}} +\left\langle-(2^n-1)\left\lfloor\frac{R_3}{2}\right\rfloor\right\rangle_{2^{n+p}}+L\right\rangle_{2^{n+p}}. \tag{15}$$

In the following equations, it is intended to simplify each term on the RHS of (15) into $(n+p)$ bits integer. This helps in reducing the hardware and delay needed to compute the value of α.

Defining

$$R_2'=\left\langle(2^n+1)\left\lfloor\frac{R_2}{2}\right\rfloor\right\rangle_{2^{n+p}}, \tag{16}$$

and

$$R_3'=\left\langle-(2^n-1)\left\lfloor\frac{R_3}{2}\right\rfloor\right\rangle_{2^{n+p}}. \tag{17}$$

The form of $\lfloor\frac{R_2}{2}\rfloor$ in n bits format, is: $\lfloor\frac{R_2}{2}\rfloor=0\,r_{2(n-1)}\cdots r_{21}$. Similarly, the form of $\lfloor\frac{R_3}{2}\rfloor$ in n bits format is: $\lfloor\frac{R_3}{2}\rfloor=r_{3n}r_{3(n-1)}\cdots r_{31}$.

The value in (16) can be expressed in binary notation as:

$$R_2'=\overbrace{r_{2p}\cdots r_{21}}^{p\ bits}\overbrace{0r_{2(n-1)}\cdots r_{21}}^{n\ bits}. \tag{18}$$

However, since:

$$\langle-2^n\lfloor\tfrac{R_3}{2}\rfloor\rangle_{2^{n+p}}=\overbrace{\overline{r}_{3p}\cdots\overline{r}_{31}}^{p}\overbrace{1\cdots1}^{n}+1$$

TABLE 1
Simplification of L_1 as Given in (13)

$r_{20}\,r_{30}$	L_1
0 0	0
0 1	$-2^{n-1}+\frac{1}{2}-\frac{2^{n-1}-\frac{1}{2}}{2^{2n}-1}$
1 0	$2^{n-1}+\frac{1}{2}+\frac{2^{n-1}+\frac{1}{2}}{2^{2n}-1}$
1 1	$1+\frac{1}{2^{2n}-1}$

$= \overbrace{\overline{r}_{3p}\cdots\overline{r}_{31}}^{p}\overbrace{0\cdots0}^{n}+2^n$, (17) can be expressed as:

$$R_3' = \overbrace{\overline{r}_{3p}\cdots\overline{r}_{31}}^{p\ bits}\overbrace{r_{3n}r_{3(n-1)}\cdots r_{31}}^{n\ bits}+2^n. \tag{19}$$

Equivalently, (19) can be rewritten as:

$$R_3' = T+2^n, \tag{20}$$

where:

$$T = \overbrace{\overline{r}_{3p}\cdots\overline{r}_{31}}^{p\ bits}\overbrace{r_{3n}r_{3(n-1)}\cdots r_{31}}^{n\ bits}. \tag{21}$$

On another track, (13) and (14) need to be simplified further and then substituted in (12) and (15).

- Simplifying L_1: According to (13), there are four cases for r_{20} and r_{30}, listed in Table 1. The table shows the result of mathematical simplification of L_1 defined in (13).
- Simplifying L_2: According to (14), simplifying L_2 depends on comparing $\lfloor\frac{R_2}{2}\rfloor$ and $\lfloor\frac{R_3}{2}\rfloor$. There are two possibilities for this comparison. The simplest form of L_2 depends also on r_{20} and r_{30} because these two bits constitute the LSB of R_2 and R_3 respectively. Simplifying L_2 should take into account the minimum and maximum difference between $\lfloor\frac{R_2}{2}\rfloor$ and $\lfloor\frac{R_3}{2}\rfloor$ because this will affect the floor value of L_1+L_2.

  Table 2 lists the two possibilities when comparing $\lfloor\frac{R_2}{2}\rfloor$ with $\lfloor\frac{R_3}{2}\rfloor$ against the four combinations of r_{20} and r_{30}, along with the minimum and maximum difference between $\lfloor\frac{R_2}{2}\rfloor$ and $\lfloor\frac{R_3}{2}\rfloor$. The corresponding simplified value of L_2 as defined in (14) against each combination is computed and shown in Table 3.

TABLE 3
Computation of $L=\lfloor L_1+L_2\rfloor$ Given in (13)–(14) Based on the Results Tabulated in Tables 1 and 2

	$r_{20}\,r_{30}$	$\lfloor L_1+L_{2_{min}}\rfloor$	$\lfloor L_1+L_{2_{max}}\rfloor$	L	$L+1$
$\lfloor\frac{R_2}{2}\rfloor \geq \lfloor\frac{R_3}{2}\rfloor$	0 0	0	0	0	1
	0 1	-2^{n-1}	-2^{n-1}	-2^{n-1}	$-2^{n-1}+1$
	1 0	2^{n-1}	2^{n-1}	2^{n-1}	$2^{n-1}+1$
	1 1	1	1	1	2
$\lfloor\frac{R_2}{2}\rfloor < \lfloor\frac{R_3}{2}\rfloor$	0 0	-1	-1	-1	0
	0 1	-2^{n-1}	-2^{n-1}	-2^{n-1}	$-2^{n-1}+1$
	1 0	2^{n-1}	2^{n-1}	2^{n-1}	$2^{n-1}+1$
	1 1	0	0	0	1

Substituting (16), (17) and the results of L listed in Table 3 in (15) leads to:

$$\alpha = \langle \overline{R}_1 + R_2' + R_3' + L + 1\rangle_{2^{n+p}}, \tag{22}$$

where the term $\langle -R_1\rangle_{2^{n+p}}$ has been replaced with its two's complement (i. e., $\langle -R_1\rangle_{2^{n+p}} = \overline{R}_1 + 1$). Hence, the binary form of $\overline{R}_1$ is given by:

$$\overline{R}_1 = \overbrace{\overline{r}_{1(n+p-1)}\cdots\overline{r}_{1n}}^{p\ bits}\overbrace{\overline{r}_{1(n-1)}\cdots\overline{r}_{10}}^{n\ bits}. \tag{23}$$

Defining w to be:

$$w = \begin{cases} 1, & \text{if } \left\lfloor\frac{R_2}{2}\right\rfloor \geq \left\lfloor\frac{R_3}{2}\right\rfloor \\ 0, & \text{if } \left\lfloor\frac{R_2}{2}\right\rfloor < \left\lfloor\frac{R_3}{2}\right\rfloor \end{cases}, \tag{24}$$

and using Table 3, then the value of $(L+1)$ given in (22) and shown in Table 4 can be expressed as:

$$L+1 = \langle 2^{n-1}a - 2^{n-1}b + c + d\rangle_{2^{n+p}}. \tag{25}$$

TABLE 2
Simplification of L_2 Given in (14)

	$r_{20}\,r_{30}$	Min. Diff. between $\lfloor\frac{R_2}{2}\rfloor$ and $\lfloor\frac{R_3}{2}\rfloor$	Value of L_2 at Min. Diff. ($L_{2_{min}}$)	Max. Diff. between $\lfloor\frac{R_2}{2}\rfloor$ and $\lfloor\frac{R_3}{2}\rfloor$	Value of L_2 at Max. Diff. ($L_{2_{max}}$)
$\lfloor\frac{R_2}{2}\rfloor \geq \lfloor\frac{R_3}{2}\rfloor$	0 0	0	0	$2^{n-1}-1$	$\frac{2^{n-1}-1}{2^n-1}$
	0 1	0	$-\frac{1}{2^{2n}-1}$	$2^{n-1}-1$	$\frac{2^{n-1}-1}{2^n-1}-\frac{1}{2^{2n}-1}$
	1 0	0	0	$2^{n-1}-2$	$\frac{2^{n-1}-2}{2^n-1}$
	1 1	0	$-\frac{1}{2^{2n}-1}$	$2^{n-1}-2$	$\frac{2^{n-1}-2}{2^n-1}-\frac{1}{2^{2n}-1}$
$\lfloor\frac{R_2}{2}\rfloor < \lfloor\frac{R_3}{2}\rfloor$	0 0	1	$-\frac{1}{2^n-1}$	-2^{n-1}	$-\frac{2^{n-1}}{2^n-1}$
	0 1	1	$-\frac{1}{2^n-1}-\frac{1}{2^{2n}-1}$	$-(2^{n-1}-1)$	$-\frac{2^{n-1}-1}{2^n-1}-\frac{1}{2^{2n}-1}$
	1 0	1	$-\frac{1}{2^n-1}$	-2^{n-1}	$-\frac{2^{n-1}}{2^n-1}$
	1 1	1	$-\frac{1}{2^n-1}-\frac{1}{2^{2n}-1}$	$-(2^{n-1}-1)$	$-\frac{2^{n-1}-1}{2^n-1}-\frac{1}{2^{2n}-1}$

TABLE 4
Expressing: $L+1=\langle 2^{n-1}a-2^{n-1}b+c+d\rangle_{2^{n+p}}$
Based on the Values of L Listed in Table 3

			$L+1$			
	r_{20}	r_{30}	a	b	c	d
$w=1$	0	0	0	0	0	1
	0	1	0	1	0	1
	1	0	1	0	0	1
	1	1	0	0	1	1
$w=0$	0	0	0	0	0	0
	0	1	0	1	0	1
	1	0	1	0	0	1
	1	1	0	0	1	0

Table 4 and (25) indicate the general form of $(L+1)$, where a, b, c, and d are single bit variables. It can be concluded from Table 4 that $a=(r_{20}\wedge\overline{r}_{30})$, $b=(\overline{r}_{20}\wedge r_{30})$, $c=(r_{20}\wedge r_{30})$, and $d=w\vee(r_{20}\oplus r_{30})$.

Substituting (25) into (22) results in:

$$\alpha=\big\langle\overline{R}_1+R'_2+R'_3+2^{n-1}a-2^{n-1}b+c+d\big\rangle_{2^{n+p}}. \tag{26}$$

In (26), adding R'_3 given in its form of (20) to $(2^{n-1}a-2^{n-1}b)$ produces:

$$R'_3+2^{n-1}a-2^{n-1}b=T+Q, \tag{27}$$

where:

$$Q=2^n+2^{n-1}a-2^{n-1}b, \tag{28}$$

a and b are fully dependent on r_{20} and r_{30}, and so is Q. Therefore, Table 5 lists all the four combinations of r_{20} and r_{30} and the corresponding value of Q as given in (28).

Moreover, Table 5 re-expresses the value of Q given in (28) as:

$$Q=t_1 2^{n-1}+t_2 2^{n-1}+t_3 2^{n-1}, \tag{29}$$

where the values of t_1, t_2 and t_3 can be concluded as: $t_1=1$, $t_2=r_{20}$ and $t_3=\overline{r}_{30}$.

Substituting (29) and (27) into (26) results in:

$$\alpha=\big\langle\overline{R}_1+R'_2+T+t_1 2^{n-1}+t_2 2^{n-1}+t_3 2^{n-1}+c+d\big\rangle_{2^{n+p}}. \tag{30}$$

Defining $Q'=t_1 2^{n-1}$, then the binary representation of Q' is:

$$Q'=\overbrace{0\cdots0}^{p\ bits}\ \overbrace{10\cdots0}^{n\ bits}. \tag{31}$$

Similarly, defining $R''_2=R'_2+t_2 2^{n-1}$, and using (18)

$$R''_2=\overbrace{r_{2p}\cdots r_{21}}^{p\ bits}\ \overbrace{r_{20}r_{2(n-1)}\cdots r_{21}}^{n\ bits}. \tag{32}$$

Moreover, defining $T'=T+t_3 2^{n-1}$, where T is given in (46), then the binary representation of T' can be put in the following form:

TABLE 5
Simplification of Q as Given in (28)

			$Q=2^n$	$Q=t_1 2^{n-1}$		
			$+a2^{n-1}$	$+t_2 2^{n-1}+t_3 2^{n-1}$,		
$r_{20}r_{30}$	a	b	$-b2^{n-1}$	t_1	t_2	t_3
0 0	0	0	2^n	1	0	1
0 1	0	1	2^{n-1}	1	0	0
1 0	1	1	2^n+2^{n-1}	1	1	1
1 1	0	0	2^n	1	1	0

$$T'=\begin{array}{l}\overbrace{\overline{r}_{3p}\cdots\overline{r}_{31}}^{p}\ \overbrace{r_{3n}r_{3(n-1)}\cdots r_{31}}^{n}\\ +\ \overbrace{0\cdots\cdots0}\ \overbrace{\overline{r}_{30}\,0\cdots\cdots\cdots0}\end{array}. \tag{33}$$

As evident from (33), r_{3n} needs to be added to $\overline{r}_{30}$. Two cases are considered here:

- Case: $r_{3n}=0$
  In this case, the sum: $r_{3n}+\overline{r}_{30}=\overline{r}_{30}$. In other words, $\overline{r}_{30}$ can replace r_{3n}. Equivalently, (33) can have the form:

$$T'=\overbrace{\overline{r}_{3p}\cdots\overline{r}_{31}}^{p}\ \overbrace{\overline{r}_{30}r_{3(n-1)}\cdots r_{31}}^{n}. \tag{34}$$

- Case: $r_{3n}=1$
  This case implies that all the remaining bits of R_3 have to be zeros, or in other words, the value of T given in (46) will be: $T=\overbrace{1\cdots11}^{p}\ \overbrace{1\,0\cdots00}^{n}$. When evaluating T' for this case:

$$T'=\begin{array}{l}\overbrace{1\cdots11}^{p}\ \overbrace{1\ 0\cdots00}^{n}\\ +\ \overbrace{0\cdots00}\ \overbrace{\overline{r}_{30}0\cdots00}\end{array}, \tag{35}$$

  in the last equation, it should be observed that the case $\overline{r}_{30}=0$, or equivalently, $r_{30}=1$ AND $r_{3n}=1$, is an impossible case. That is, for the case: $r_{3n}=1$, the value of $\overline{r}_{30}$ has to be 1. Therefore, the last equation can be written as:

$$T'=\begin{array}{l}\overbrace{1\cdots11}^{p}\ \overbrace{1\,0\cdots00}^{n}\\ +\overbrace{0\cdots00}\ \overbrace{10\cdots00}\end{array}, \tag{36}$$

  which turns the value of T' to 0:

$$\langle T'\rangle_{2^{n+p}}=\ \overbrace{0\cdots00}^{p}\ \overbrace{0\,0\cdots00}^{n}. \tag{37}$$

Combining both cases of T' given in (34) and (37) in one variable denoted as R''_3, then:

$$R''_3=\overbrace{(\overline{r}_{3n}\wedge\overline{r}_{3p})\cdots(\overline{r}_{3n}\wedge\overline{r}_{31})}^{p}\ \overbrace{(\overline{r}_{3n}\wedge\overline{r}_{30})\,r_{3(n-1)}\cdots r_{31}}^{n}. \tag{38}$$

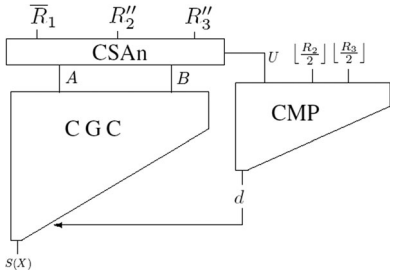

Fig. 1. Block diagram of the proposed sign detector.

Therefore, the expression for α in (30) can be rewritten as:

$$\alpha \quad = \langle \overline{R}_1 + R''_2 + R''_3 + Q' + c + d \rangle_{2^{n+p}}, \tag{39}$$

where: $\overline{R}_1$, R''_2 and R''_3 are given in (23), (32) and (38), respectively, while $Q' = 2^{n-1}$, as given in (31). However, c and d were defined before to be: $c = (r_{20} \wedge r_{30})$, and $d = w \vee (r_{20} \oplus r_{30})$, while, w was introduced in (24).

Sign Detection Algorithm: Based on (39), the sign detection algorithm proposed in paper can be summarized as follows:

Given the moduli set $(2^{n+p}, 2^n - 1, 2^n + 1)$ where n and p are positive integers such that $n > p$. Given $X \in [0, M)$, expressed as: (R_1, R_2, R_3). The sign of X can be computed as follows:

- Formulate $\overline{R}_1$, R''_2 and R''_3 .
- Compute c and d.
- Given $Q' = 2^{n-1}$, compute:
  $\alpha = \langle \overline{R}_1 + R''_2 + R''_3 + Q' + c + d \rangle_{2^{n+p}}$
- $S(X) = \alpha_{n+p-1}$, i. e., the MSB of α.

Numerical Example 1: Given the moduli set $(128, 31, 33)$, determine the sign of the residue number $X = (80, 14, 27)$. $n = 5$, $p = 2$, $n + p = 7$. In binary form, $R_1 = 1010000$, $R_2 = 01110$ and $R_3 = 011011$. Consequently, $\overline{R}_1 = \ 01\ 01111 \overset{decimal}{\longrightarrow} 47$, $R''_2 = 11\ 00111 \overset{decimal}{\longrightarrow} 103$, $R''_3 = 10\ 01101 \overset{decimal}{\longrightarrow} 77$ and $Q' = 16$. Similarly, because $\left\lfloor \frac{R_2}{2} \right\rfloor < \left\lfloor \frac{R_3}{2} \right\rfloor$, $r_{20} = 0$ and $r_{30} = 1$ then $w = 0$, $c = 0$ and $d = 1$. Applying (39), $\alpha = \langle 47 + 103 + 77 + 16 + 0 + 1 \rangle_{128} = 116 \overset{binary}{\longrightarrow} \underline{1}110100$. $S(X) = x_7 = 1$, therefore, X is negative. Actually, $X = 118,992 \in [\frac{M}{2}, M)$.

Considering another case for which $r_{3n} = 1$. For the same previous moduli set $(128, 31, 33)$, determine the sign of the residue number $(80, 14, 32)$. The only difference in this example is: $R''_3 = 0$, $d = 0$. Therefore, $\alpha = \langle 47 + 103 + 0 + 16 + 0 + 0 \rangle_{128} = 38 \overset{binary}{\longrightarrow} \underline{0}010110$. $S(X) = \alpha_7 = 0$, therefore, X is positive. Actually, $X = 39,632 \in [0, \frac{M}{2})$.

Numerical Example 2: Given the moduli set $(2^{20}, 2^{16} - 1, 2^{16} + 1)$, where $n = 16$, $p = 4$, $n + p = 20$. We need to determine the sign of the residue number $X = (920519, 61500, 42328)$. In binary form,
$R_1 = \underline{1110}\ \underline{0000}\ \underline{1011}\ \underline{1100}\ \underline{0111}$,
$R_2 = \underline{1111}\ \underline{0000}\ \underline{0011}\ \underline{1100}$, and
$R_3 = 0\ \underline{1010}\ \underline{0101}\ \underline{0101}\ \underline{1000}$. Therefore,
$\overline{R}_1 = \underline{0001}\ \underline{1111}\ \underline{0100}\ \underline{0011}\ \underline{1000} \overset{decimal}{\longrightarrow} 128056$,
$R''_2 = \underline{1110}\ \underline{0111}\ \underline{1000}\ \underline{0001}\ \underline{1110} \overset{decimal}{\longrightarrow} 948254$,
$R''_3 = \underline{0011}\ \underline{1101}\ \underline{0010}\ \underline{1010}\ \underline{1100} \overset{decimal}{\longrightarrow} 250540$, and $Q' = 2^{15}$. Similarly, because $\lfloor \frac{R_2}{2} \rfloor \geq \lfloor \frac{R_3}{2} \rfloor$, $r_{20} = 0$ and $r_{30} = 0$, then, $w = 1$, $c = 0$ and $d = 1$. Applying (39), $\alpha = \langle 128056 + 948254 + 250540 + 2^{15} + 0 + 1 \rangle_{2^{20}} = 311043 \overset{binary}{\longrightarrow} \underline{0100}\ \underline{1011}\ \underline{1111}\ \underline{0000}\ \underline{0011}$. Hence, $S(X) = \alpha_{19} = 0$. This implies that X is positive. Actually , $X = (1\ 335\ 920\ 140\ 618\ 695) \in [0, \frac{M}{2})$, where $\frac{M}{2} = \frac{2^{52} - 2^{20}}{2} = 2\ 251\ 799\ 813\ 160\ 960$.

For the same moduli set, we need to determine the sign of the residue number $X = (1048575, 65534, 65536)$, In binary form,
$R_1 = \underline{1111}\ \underline{1111}\ \underline{1111}\ \underline{1111}\ \underline{1111}$,
$R_2 = \underline{1111}\ \underline{1111}\ \underline{1111}\ \underline{1110}$, and
$R_3 = 1\ \underline{0000}\ \underline{0000}\ \underline{0000}\ \underline{0000}$. Therefore,
$\overline{R}_1 \overset{decimal}{\longrightarrow} 0$, $R''_2 \overset{decimal}{\longrightarrow} 1015807$, $R''_3 = \overset{decimal}{\longrightarrow} 0$, and $Q' = 2^{15}$. Similarly, because $\left\lfloor \frac{R_2}{2} \right\rfloor < \left\lfloor \frac{R_3}{2} \right\rfloor$, $r_{20} = 0$ and $r_{30} = 0$, then, $w = 0$, $c = 0$ and $d = 0$. Applying (39), $\alpha = \langle 0 + 1015807 + 0 + 2^{15} + 0 + 0 \rangle_{2^{20}} = 1048575 \overset{binary}{\longrightarrow} \underline{1111}\ \ \underline{111}\ \underline{1111}\ \underline{1111}\ \underline{1111}$. Hence, $S(X) = \alpha_{19} = 1$. This means that X is negative. In fact, $X = (4\ 503\ 599\ 626\ 321\ 919) \in [\frac{M}{2}, M)$.

5 Proposed Hardware Design

5.1 Sign Detector Design for $(2^{n+p}, 2^n - 1, 2^n + 1)$, General Case

Implementing (39) basically requires adding $\overline{R}_1$, R''_2, R''_2 and Q' while incorporating the two single bits: c and d.

Fig. 1 shows the block diagram for the proposed sign detector. It consists of a Carry-Save Adder network (CSAn), a Carry Generation Circuit (CGC) and a Comparator circuit (CMP). The block diagram operates in the following manner:

- The CSAn adds $\overline{R}_1$, R''_2, R''_2 and Q', where each consists of $(n + p)$ bits. It produces two vectors: the sum vector: $A = A_{n+p-1} \cdots A_1 A_0$, and the carry vector: $B = B_{n+p-1} \cdots B_1 0$, where the single bit variable c given in (39) can be superimposed on B because the LSB of B is 0, which leads to: $B + c = B_{n+p-1} \cdots B_1 c$. Any more significant bits A_i or B_i for $i \geq (n + p)$ are ignored because (39) applies modulus 2^{n+p}.

  Fig. 2 shows the detailed structure of the CSAn for the case: $n = 8$ and $p = 4$. This CSAn requires

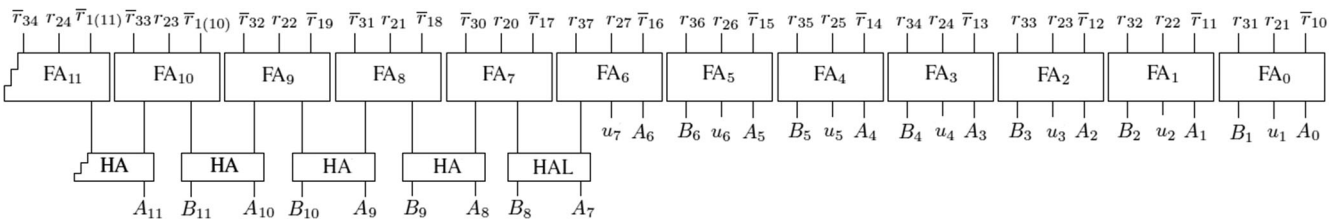

Fig. 2. The detailed design of the CSAn structure for the case $p = 4$ and $n = 8$.

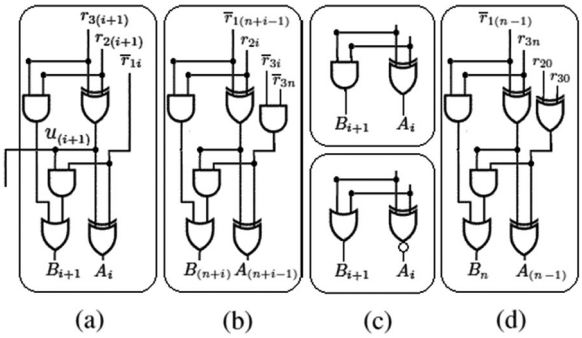

Fig. 3. The components used in the CSAn: (a) The regular FAs used for each FA_i, $0 \leq i < (n-1)$, to generate $u_{(i+1)}$ (b) the customized FAs used for each $FA_{(n+i-1)}$, $0 \leq i < p$, (c) the regular HA and HAL designs, (d) FA_{n-1} used for the case $p = 1$.

$(n+p-1)$ Full-Adders (FAs) and p Half Adders (HA)/ Half-Adder-Like circuits (HALs). The HAL circuit is a minimized form of the FA circuit in which the third input is 1. This is needed in the CSAn to incorporate the non-zero bit in Q'. The functionality of the HAL circuit which accepts two bits and produces the sum and carry bits is realized by two gates, namely: XNOR and OR gates. Fig. 3a shows the structure of the regular FA_i that is used for $0 \leq i < (n-1)$. Fig. 3b shows the customized FA_i that is used for $(n-1) \leq i < (n+p-1)$ which incorporates the bit $\overline{r}_{3n} \wedge \overline{r}_{3i}$ given in (38). Fig. 3c shows the structure of the HA and HAL circuits.

Without additional cost, the LS $(n-1)$ FAs of the CSAn can be used to produce the by-product vector $U = u_n u_{n-1} \cdots u_1$, where $u_i = r_{2i} \oplus r_{3i}$, for $1 \leq i \leq n$. This will eliminate the need to generate the same vector in the CMP circuit. This approach has been introduced in [9]. The MSB of U, namely, $u_n = r_{3n} \oplus 0 = r_{3n}$ is available.

- The CMP circuit receives $\lfloor \frac{R_2}{2} \rfloor$, $\lfloor \frac{R_3}{2} \rfloor$ and U in the form of n bits each, where the MSB of $\lfloor \frac{R_2}{2} \rfloor$ is 0. The purpose of this circuit is to determine if $\lfloor \frac{R_2}{2} \rfloor \geq \lfloor \frac{R_3}{2} \rfloor$ (i. e., that is to compute the output carry resulting from $(\lfloor \frac{R_2}{2} \rfloor - \lfloor \frac{R_3}{2} \rfloor)$, namely w, and consequently compute the value of d given in (39)). This can be done using 2's complement notation as: $\lfloor \frac{R_2}{2} \rfloor - \lfloor \frac{R_3}{2} \rfloor = \lfloor \frac{R_2}{2} \rfloor + \overline{\lfloor \frac{R_3}{2} \rfloor} + 1$. The output carry resulting from this subtraction will be w. Alternatively, and to get red of the $+1$ in the last expression, the comparison can be done using 1's complement notation as: $\lfloor \frac{R_3}{2} \rfloor - \lfloor \frac{R_2}{2} \rfloor = \lfloor \frac{R_3}{2} \rfloor + \overline{\lfloor \frac{R_2}{2} \rfloor}$. The output carry resulting from this case will be $\overline{w}$. We have chosen to adopt the latter approach. Fig. 4 shows the detailed design of the CMP circuit for the case $n = 8$, while Fig. 5 shows the gate-level design of the basic cells of the parallel-prefix networks used in this paper [17], [18], [19], [20], [21], [22], [23].

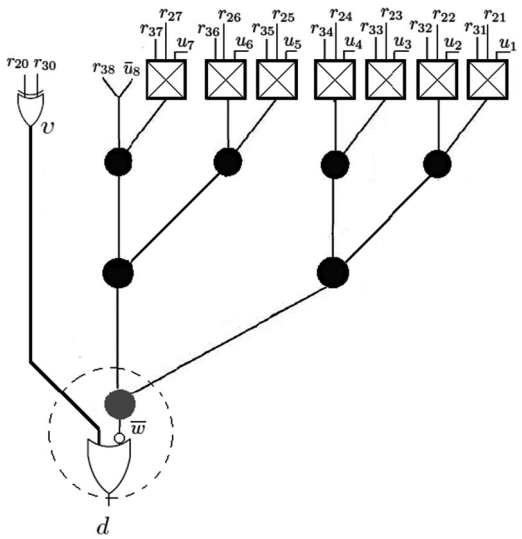

Fig. 4. Design of the comparator circuit for the case $n = 8$, where the components enclosed within the dashed circle can be replaced with a hollow node (shown in Fig. 5).

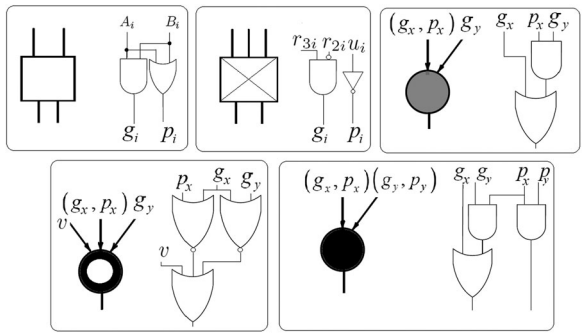

Fig. 5. Gate-level structure of basic cells used in this work for parallel-prefix networks.

The very bottom node in Fig. 4 is a gray node (a gray node is used instead of a black one because one of the AND gate of the black node is not needed at this stage). The output carry coming out from the OR gate of the gray node in Fig. 4 is given by: $\overline{w} = g_x \vee (p_x \wedge g_y)$, where g_x, p_x and g_y are the inputs of the node. The output of the OR gate that follows the bottom gray node in Fig. 4, is given by: $d = v + w$, where $v = (r_{20} \oplus r_{30})$. Equivalently, $d = v \vee \overline{(g_x \vee (p_x \wedge g_y))}$. Using Boolean Algebra, the expression for d is: $d = v \vee (\overline{g_x} \wedge \overline{p_x}) \vee (\overline{g_x} \wedge \overline{g_y})$. This implies that the bottom gray node followed with the OR gate which are enclosed within the dashed circle in Fig. 4 can be replaced with the hollow node shown in Fig. 5.

- The CGC receives the two vectors A and $B + c$. Fig. 6 shows the detailed structure of this circuit for the case $n = 8$ and $p = 4$. The bottom black node in the parallel-prefix network shown in Fig. 6 produces an output carry. This carry results from adding the LS $(n+p-1)$ bits of A and $B + c$.

The gray node incorporates the value of d, similar to an end-around carry [17], [18], [19], [20], [21], [22], [23]. The XOR gate following the gray node produces the MSB of $S(X)$.

Similar to what have been done in the CMP circuit, the gray node and the black node that precedes it, enclosed within the dashed circle in Fig. 6, can be simplified as follows: The inputs to the bottom black node in Fig. 6 can be labeled as: g_x, p_x, g_y and p_y. The corresponding outputs coming from the same black

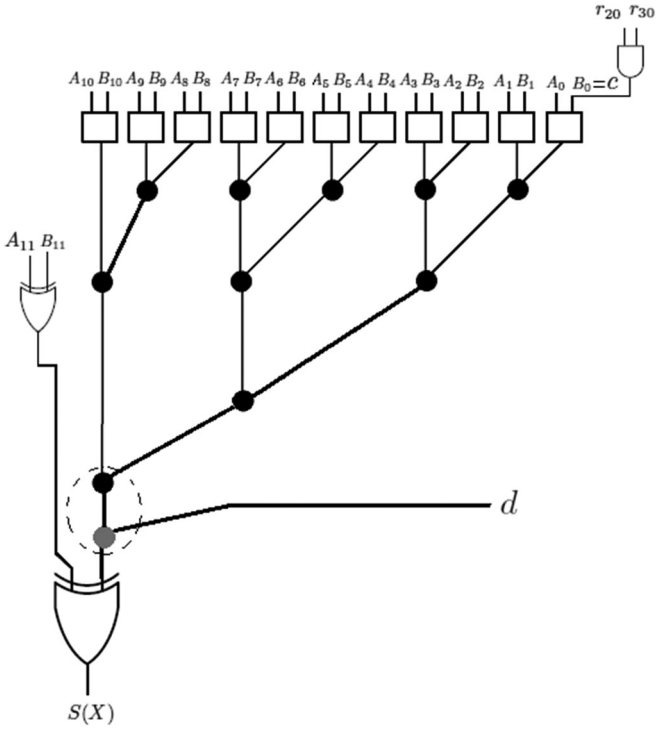

Fig. 6. Design of the CGC for the case: $n+p=12$ ($n=8$ and $p=4$), where the components enclosed within the dashed circle can be replaced with the circuit shown in Fig. 7.

node are given by: g_z and p_z, respectively, where $g_z = g_x \vee (p_x \wedge g_y)$ and $p_z = p_x \wedge p_y$. The inputs to the gray node are: g_z, p_z and d, where the output of this gray node is given by a variable g, where $g = g_z \vee (p_z \wedge d)$. Upon substituting the values of g_z and p_z, it can be concluded that: $g = g_x \vee (p_x \wedge g_y) \vee (p_x \wedge p_y \wedge d)$. The hardware implementation of the last expression is shown in Fig. 7.

5.2 Sign Detector Design for $(2^n, 2^n-1, 2^n+1)$, $p=0$

In this section, we are customizing the previous analysis needed to compute α for this specific case, of $p=0$, in order to minimize the hardware and delay requirements. In specific, we will eliminate the need for HA/HAL circuit in the CSAn. That is, the CSAn will require only the first level of FAs.

Applying modulus 2^n when calculating α, then (28) can be written as:

$$\langle Q \rangle_{2^n} = \langle 2^{n-1}a + 2^{n-1}b \rangle_{2^n}. \tag{40}$$

Since $a = \overline{r}_{20} \wedge r_{30}$, and $b = r_{20} \wedge \overline{r}_{30}$, the last equation leads to:

$$Q = 2^{n-1}(r_{20} \oplus r_{30}), \tag{41}$$

and since $\langle R'_3 \rangle_{2^n} = \langle T + 2^n \rangle_{2^n} = T$, then, (26) can be rewritten for this case as:

$$\begin{aligned} \alpha = \big\langle \overline{R}_1 + R'_2 + R'_3 \\ + 2^{n-1}(r_{20} \oplus r_{30}) + c + d \big\rangle_{2^n}, \end{aligned} \tag{42}$$

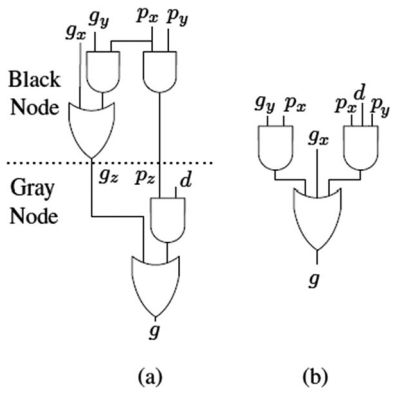

Fig. 7. Design of the circuit that can replace the components enclosed within the dashed circle in Fig. 6: (a) the nodes before being combined, (b) the nodes after being combined into one node.

where the term $2^{n-1}(r_{20} \oplus r_{30})$ in the last equation can be superimposed on the value of R'_2 (i. .e replacing the zero bit in R'_2 given in (18) whose binary weight is 2^{n-1}) to produce R''_2 as follows:

$$\alpha = \big\langle \overline{R}_1 + R''_2 + R'_3 + c + d \big\rangle_{2^n}, \tag{43}$$

where:

$$\overline{R}_1 = \overbrace{\overline{r}_{1(n-1)} \cdots \overline{r}_{10}}^{n\ bits} \tag{44}$$

$$R''_2 = \overbrace{(r_{20} \oplus r_{30})\, r_{2(n-1)} \cdots r_{21}}^{n\ bits} \tag{45}$$

$$R'_3 = \overbrace{r_{3n} r_{3(n-1)} \cdots r_{31}}^{n\ bits}. \tag{46}$$

The sign of X is given by the MSB of α, that is,

$$S(X) = \alpha_{n-1}. \tag{47}$$

Examining (43)-(46) reveals that the MSB of the sum vector $A_{n-1} = \overline{r}_{1(n-1)} \oplus (r_{20} \oplus r_{30}) \oplus r_{3n}$, can be computed in parallel with the parallel prefix network needed to compute the carry resulting from adding $A_{n-2} \cdots A_0$ and $B_{n-2} \cdots B_1 c$. Thus, it adds no delay.

5.3 Sign Detector Design for $(2^{n+1}, 2^n-1, 2^n+1)$, $p=1$

As have been done in the previous section, the generalized results need to be customized to eliminate HA/HALs circuits in the CSAn for the case $p=1$ and thus reduce hardware and time requirements.

Recalling that modulus 2^{n+1} needs to be applied in calculating α, then (28) can be written as:

$$\langle Q \rangle_{2^{n+1}} = \langle 2^n + 2^{n-1}a - 2^{n-1}b \rangle_{2^{n+1}}. \tag{48}$$

Reconsidering Table 5, then for the case $p=1$, this table can be re-structured in a new form shown in Table 6, where the latter leads to expressing $Q = t'_1 2^n + t'_2 2^{n-1}$, where $t'_1 = r_{20} \vee \overline{r}_{30}$ and $t'_2 = r_{20} \oplus r_{30}$. Therefore, (48) can be written as:

TABLE 6
Simplification of Q Given in (48) for the Case $p = 1$

$r_{20}\,r_{30}$	$a\;\;b$	$Q = 2^n + a2^{n-1} - b2^{n-1}$	$Q = t'_1 2^n + t'_2 2^{n-1}$	
			t'_1	t'_2
0 0	0 0	2^n	1	0
0 1	0 1	2^{n-1}	0	1
1 0	1 1	$2^n + 2^{n-1}$	1	1
1 1	0 0	2^n	1	0

$$Q = (r_{20} \vee \overline{r}_{30})2^n + (r_{20} \oplus r_{30})2^{n-1}. \tag{49}$$

In (49), the term $2^{n-1}(r_{20} \oplus r_{30})$ will be superimposed on the value of R'_2 to produce R''_2. However, the term $(r_{20} \vee \overline{r}_{30})$ needs to be added to $\overline{r}_{31}$ in R'_3 to produce R''_3. That is, $\langle \overline{r}_{31} + (r_{20} \vee \overline{r}_{30}) \rangle_{2^{n+1}} = r_t$, where $r_t = \overline{r}_{31} \oplus (r_{20} \vee \overline{r}_{30})$.

Reflecting the above conclusions on the value of α given in (26), produces:

$$\alpha = \langle \overline{R}_1 + R''_2 + R''_3 + c + d \rangle_{2^{n+1}} \tag{50}$$

$$\overline{R}_1 = \overbrace{\overline{r}_{1n}}^{1}\overbrace{\overline{r}_{1(n-1)} \cdots \overline{r}_{10}}^{n\ bits} \tag{51}$$

$$R''_2 = \overbrace{r_{21}}^{1}\overbrace{(r_{20} \oplus r_{30}) r_{2(n-1)} \cdots r_{21}}^{n\ bits}. \tag{52}$$

$$R''_3 = \overbrace{r_t}^{1}\overbrace{r_{3n}\, r_{3(n-1)} \cdots r_{31}}^{n\ bits}. \tag{53}$$

The sign bit is then given by:

$$S(X) = \alpha_n. \tag{54}$$

Computing $(r_{20} \oplus r_{30})$ in (52) does not introduce any delay since it can be performed as shown in Fig. 3d, where the full adder structure incorporates the expression $(r_{20} \oplus r_{30})$. Examining (50)-(53) also indicates that the MSB of the sum vector $A_n = \overline{r}_{1n} \oplus r_{21} \oplus r_t$, can be computed in parallel with the parallel-prefix network that produces $S(X)$. Thus, it adds no delay.

6 VLSI Realization and Comparative Analysis

The structures of sign detectors that depend on ROMs and multipliers [4], [5], [6] are not competitive with the ones that depend mainly on adders [7], [8], [9], [10], [11], [12], [13], [14], especially for large values of n. The magnitude comparator in [11] was found to be very demanding if customized to function as a sign detector.

Therefore, and in order to get a better realistic estimation for the new sign detector, the structure proposed in this paper and the structures of the most competitive published work [7], [8], [9], [10] and [12], [13], [14] have all been modeled using Verilog HDL and synthesized using Synopsys Design Compiler (Version G-2012.06) by mapping the design to 65 nm Synopsys DesignWare Logic Libraries. The functionality correctness of each sign detector under consideration has been checked using Synopsys Simulator.

TABLE 7
Area, Delay and Power for the Proposed Sign Detector and Other Competitive Work

Structure	n	p	Area μm^2	Delay ps	Power μW
Proposed	8	0	501.2	461.4	32.0
		1	553.4	468.3	34.6
		4	739.7	511.1	42.4
Proposed	16	0	998.6	515.3	46.4
		1	1,043.9	522.4	48.3
		8	1,417.6	564.6	55.8
[10]	8	0	811.6	756.1	42.7
		1	922.8	769.4	44.5
		4	1,051.4	778.5	50.8
[10]	16	0	1,740.4	834.3	64.3
		1	1,890.2	839.2	66.5
		8	2,178.0	861.6	71.2
[7]	8	0	617.2	539.1	38.1
	16	0	1,145.3	589.9	51.9
[8]	8	0	887.5	673.8	45.2
	16	0	1,634.3	728.2	63.1
[9]	8	0	535.9	521.2	34.1
	16	0	1,059.8	577.8	48.7
[12]	8	1	645.4	534.7	39.8
	16	1	1,181.5	591.5	54.0
[13]	8	1	832.7	654.4	44.7
	16	1	1,505.6	701.3	58.5
[14]	8	1	867.3	667.9	45.1
	16	1	1,573.5	720.8	61.9

Synopsys Power Compiler was also used to estimate the power consumption. The structures under consideration were recursively optimized for speed until no more improvement could be obtained. Table 7 summarizes the synthesis results of the area, delay and power of the proposed sign detector for the moduli set $(2^{n+p}, 2^n - 1, 2^n + 1)$ for different values of n and p. Table 7 also lists area, delay and power of the sign detectors in [7], [8], [9], [10] and [12], [13], [14] using the same tools. The reductions in area, delay and power of the new proposed sign detector as concluded from Table 7 have been reported in Table 8.

The proposed work and the work in [10] both deal with the same moduli set, namely, $(2^{n+p}, 2^n - 1, 2^n + 1)$, which is the main focus of this paper. However, when the proposed structure is compared with that in [10], the proposed one has achieved notable large reductions in area given by the range (29.6-44.8 percent), in delay given by the range (34.3-39.1 percent) and in power given by the range (16.6-27.8 percent).

When comparing the new structure (for $p = 0$) with the one in [7], the new design has achieved a reduction in area given by the range (12.8-18.8 percent), in delay given by the range (12.6-14.4 percent) and in power given by the range (10.6-16.0 percent). When the new structure (for the case $p = 0$) is compared with that in [8], the proposed one has a reduction in area given by the range (38.9-43.5 percent)), and in delay given by the range (29.2-31.5 percent) and in power given by the range (26.5–29.2 percent). When comparing the proposed circuit (for $p = 0$) with that in [9], Table 8 shows improvements in area in the range of (5.3-6.5 percent), in delay in the range of (10.8-11.5 percent) and in power in the range of (4.7-6.2 percent).

TABLE 8
Area, Delay and Power Savings of the Proposed Sign Detector as Compared with Other Competitive Work

Proposed compared with:	n	p	(%) Area Reduction	(%) Delay Reduction	(%) Power Reduction
		0	38.2	39.0	25.1
[10]	8	1	40.0	39.1	22.2
		4	29.6	34.3	16.6
		0	42.6	38.2	27.8
[10]	16	1	44.8	37.8	27.4
		8	34.9	34.5	21.6
[7]	8	0	18.8	14.4	16.0
	16	0	12.8	12.6	10.6
[8]	8	0	43.5	31.5	29.2
	16	0	38.9	29.2	26.5
[9]	8	0	6.5	11.5	6.2
	16	0	5.3	10.8	4.7
[12]	8	1	14.3	12.4	13.1
	16	1	11.6	11.7	10.6
[13]	8	1	33.5	28.4	22.6
	16	1	30.7	25.5	17.4
[14]	8	1	36.2	29.9	23.3
	16	1	33.7	27.5	22.0

However, when compared with the structure in [12], the proposed design (for $p = 1$) has achieved area improvements in the range of (11.6-14.3 percent), delay improvements in the range of (11.7-12.4 percent) and power improvements in the range of (10.6–13.1 percent). When the proposed sign detector (for the case $p = 1$) is compared with that in [13], the new structure has achieved a reduction in area given by the range (30.7-33.5 percent)), and in delay given by the range (25.5-28.4 percent) and in power given by the range (17.4-22.6 percent). Similar conclusions can be made when comparing the new sign detector ($p = 1$) with the one in [14]. The new one has reduced area , delay and power by (33.7-36.2 percent), (27.5-29.9 percent) and (22.0-23.3 percent) respectively.

7 Conclusions

This paper introduced a new structure for a sign detector for the group of moduli sets that has the form $(2^{n+p}, 2^n - 1, 2^n + 1)$, where n and p are positive integers such that $p < n$. The proposed design is flexible and can easily deal with all dynamic ranges starting from $3n$ bits and up to $3n + p$ bits dynamic ranges. For cases where $p > 1$, increasing the dynamic range by 1 bit (i. e., doubling the dynamic range) requires only one half-adder and no additional delay. When compared with the most competitive published work, the new sign detector exhibits a better performance in terms of area, delay and power that ranges from 4.7 percent up to 44.8 percent.

Acknowledgments

The author would like to thank the anonymous reviewers for their constructive comments and suggestions that significantly contributed to improving the paper.

References

[1] N. Szabo, and R. Tanaka, *Residue Arithmetic and Its Applications to Computer Technology*. New York, NY, USA: McGraw Hill, 1967.
[2] M. Soderstrand, M. A., W. Jenkins, G. Jullien, F. Taylor, Eds., *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*. New York, NY, USA: IEEE Press, 1986.
[3] P.V. Ananda Mohan, *Residue Number Systems: Algorithms and Architectures*. Berlin, Germany: Springer, 2002.
[4] Z. D. Ulman, "Sign detection and implicit-explicit conversion of numbers in residue arithmetic," *IEEE Trans. Comp.*, vol. 32, no. 6, pp. 590–594, Jun. 1983
[5] T. V. Vu, "Efficient implementations of Chinese remainder theorem for sign detection and residue decoding," *IEEE Trans. Comp.*, vol. 34, no. 7, pp. 646–651, Jul. 1985.
[6] G. Alia and E. Martinelli, "Sign detection in residue arithmetic units," *J. Syst. Arch.*, vol. 45, no. 2, pp. 251–258, 1998.
[7] T. Tomczak, "Fast sign detection for RNS $(2^n - 1, 2^n, 2^n + 1)$," *IEEE Trans. Circuits Syst. I*, vol. 55, no. 6, pp. 1502–1511, Jul. 2008.
[8] Y. Wang, X. Song, M. Aboulhamid, and H. Shen, "Adder-based residue to binary number converter for $(2^n - 1, 2^n, 2^n + 1)$," *IEEE Trans. Signal Process.*, vol. 50, no. 7, pp. 1772–1779, Jul. 2002.
[9] M. Xu, R. Yao and F. Luo, "Low-complexity sign detection algorithm for RNS $\{2^n - 1, 2^n, 2^n + 1\}$," *IEICE Trans. Electron.*, vol. E95.C, pp. 1552–1556, Sep. 2012.
[10] L. Sousa and P. Srgio Alves Martins, "Efficient sign identification engines for integers represented in the RNS extended 3-moduli set $\{2^n - 1, 2^{n+k}, 2^n + 1\}$," *Electron. Lett.*, vol. 50, n. 16, pp. 1138–1139, Jul. 2014.
[11] G. Pirlo and S. Impedovo, "A new class of monotone functions of the residue number system," *Int. J. Math. Models Methods Appl. Sci.*, vol. 7, no. 9, pp. 803–809, 2013.
[12] M. Xu, Z. Bian, and R. Yao, "Fast sign detection algorithm for the RNS moduli set $\{2^{n+1} - 1, 2^n - 1, 2^n\}$," *IEEE Trans. VLSI Syst.*, vol. 23, no. 2, pp. 379–383, Feb. 2015.
[13] E. Al-Radadi and P. Siy, "RNS sign detector based on Chinese remainder theorem II (CRT II)," *Comput. Math. Appl.*, vol. 46, no. 10–11, pp 1559–1570, Nov. 2003.
[14] M. Akkal and P. Siy, "Optimum RNS sign detection algorithm using MRC-II with special moduli set," *J. Syst. Arch.*, vol. 54, no. 10, pp. 911–918, Oct. 2008.
[15] T. F. Tay, C. H. Chang and L. Sousa, "Base transformation with injective residue mapping for dynamic range reduction in RNS," *IEEE Trans. Circuits Syst. I*, vol. 62, no. 9, pp. 2248–2259, Sep. 2015.
[16] A. Sweidan, and A. Hiasat, "New efficient memoryless, residue to binary converter," *IEEE Trans. Circuits Syst.*, vol. 35, no. 11, pp. 1441–1444, Nov. 1988.
[17] R. Zimmermann, "Efficient VLSI implementation of modulo $(2^n \pm 1)$ addition and multiplication," in *Proc. 14th IEEE Symp. Comput. Arithmetic*, Apr. 14–16, 1999, pp. 158–167.
[18] G. Dimitrakopoulos and D. Nikolos, "High-speed parallel-prefix VLSI ling adders," *IEEE Trans. Comput.*, vol. 54, no. 2, pp 225–231, Feb. 2005.
[19] R. A. Patel, M. Benaissa, and 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, Nov. 2007.
[20] G. Jaberipur and B. Parhami, "Unified approach to the design of modulo- $(2^n \pm 1)$ adders based on signed-LSB representation of residues," in *Proc. 19th IEEE Symp. Comput. Arithmetic*, pp. 57–64, 2009.
[21] H. T. Vergos and G. Dimitrakopoulos, "On modulo $2^n + 1$ adder design," *IEEE Trans. Comput.*, vol. 61, no. 2, pp. 173–186, Feb. 2012.
[22] S. Ma, J. H. Hu, and C. H. Wang, "A novel modulo $2^n - 2^k - 1$ adder for residue number system," *IEEE Trans. Circuits Syst. I*, vol. 60, no. 11, pp. 2962–2972, Oct. 2013.
[23] R. Muralidharan and C. H. Chang, "Area-power efficient modulo $2^n - 1$ and modulo $2^n + 1$ multipliers for $\{2^n - 1, 2^n, 2^n + 1\}$ based RNS," *IEEE Trans. Circuits Syst. I*, vol. 59, no. 10, pp. 2263–2274, Oct. 2012.

Ahmad Hiasat received the BSc and MSc degrees in electrical engineering from the University of Jordan, Amman, Jordan. He received the PhD degree in systems engineering from Oakland University, MI, USA in 1995. He then joined Princess Sumaya University for Technology (PSUT). He has been a full professor since 2005. Seconded from PSUT, he served as the Chairman and CEO of Telecommunications Regulatory Commission of Jordan during 2006-2010 and the Chairman and CEO of Energy Regulatory Commission of Jordan during 2011-2012. His research interests include; computer arithmetic, residue number system, digital median filters, and VLSI design.