Universal Approaches for Overflow and Sign Detection in Residue Number System Based on $\{2^n - 1, 2^n, 2^n + 1\}$

Dina Younes, Pavel Steffan
Dept. of Microelectronics
Brno University of Technology
Brno, Czech Republic
xyoune00@stud.feec.vutbr.cz, steffan@feec.vutbr.cz

Abstract—**This paper presents two universal efficient approaches for overflow and sign detection and correction in the addition of two numbers in unsigned and signed residue number systems (RNS). Both methods are designed to be used in systems based on the moduli set $\{2^n - 1, 2^n, 2^n + 1\}$ that provides an even dynamic range. Moreover, by applying a tiny modification, these designs can be used in any system that has (2^n) as one of its moduli (i.e. has an even dynamic range). The proposed methods depend on a simple structure that provides fast and accurate detection and correction of the sign and overflow. A comparison, which proves the efficiency of the proposed designs, in terms of time and area requirements is also presented.**

Keywords-residue number system (RNS); overflow detection; sign detection; even dynamic range; moduli set $\{2^n - 1, 2^n, 2^n + 1\}$

I. INTRODUCTION

The residue number system (RNS) is a unique, non-weighted, carry-free number system that provides parallel, high speed and fault tolerant arithmetic operations. This makes it a tough candidate for high-performance, low power, fault tolerant and secure digital signal processing (DSP) applications. This system has been intensively used in applications where addition, subtraction and multiplication are dominant, such as, digital filters, digital communications, discrete Fourier transform (DFT), image processing and video coding [1], [2].

Nevertheless, operations as division, overflow detection, sign detection and magnitude comparison are problematic and very complex in RNS. In some cases, some of these operations, such as overflow and sign detection, are essential and cannot be avoided. Moreover, they are fundamental in other operations such as division.

Sign and overflow detection are very important issues in RNS, since a wrong detection of sign or overflow ruins the whole advantage of using RNS. There is no point of using RNS in order to obtain parallel, high-speed and secure arithmetic operations if the results of these operations are wrong.

In principle, the general way to detect overflow in RNS is via comparing the result of addition with one of the addends. If $X \geq 0$ and $Y < M$ then $(X+Y) \bmod M$ causes overflow if and only if the result is less than X.

On the other hand, the general way for sign detection in RNS is via comparing the converted number from residue-to-binary with half of the dynamic range of the RNS. Thus, we can conclude, that both sign detection and overflow detection are equivalent to the magnitude comparison.

One of the most efficient ways to detect overflow in RNS is via parity checking [1], [2], [3], and [4]. It indicates whether an integer is even or odd. Suppose two integers (X, Y) have the same parity: $Z = X + Y$. An overflow occurs if Z is odd. Contrary, if (X, Y) have different parity, then an overflow occurs if Z is even. The parity checking technique is one of the best and fastest suggested methods to detect the overflow in RNS. It depends on look-up tables (LUTs) or on an extra modulo (a redundant modulo). However, this technique can only be used with moduli sets that have just odd members, i.e. odd dynamic range, which is not suitable for many moduli sets that uses 2^n as one of its moduli, especially the most famous moduli set $\{2^n - 1, 2^n, 2^n + 1\}$. RNS systems, that have even dynamic ranges, have more attractive features than those with odd dynamic ranges. Due to the reason, that using (2^n) modulo greatly simplifies and reduces the delay and complexity of the residue arithmetic operations and the residue-to-binary conversion.

Thus, it is obvious that overflow detection in RNS that has an even dynamic range is a very important issue. As aforementioned before, both sign detection and overflow detection, based on the general ways, are equivalent to the magnitude comparison. Therefore, many RNS comparators have been presented [5], [6], and [7]. They used many different techniques in order to obtain faster performance and smaller area consumption. In [6], a residue comparator based on the Chinese reminder theorem (CRT) for general modulo sets is presented. In [7], a comparator based on diagonal function, that is named SUM of Quotients Technique (SQT), is introduced. An efficient residue comparator for any odd moduli set, which is based on the parity of integers and their period, is stated in [5].

In this paper, we present two efficient techniques for sign and overflow detection and correction in RNS based on the moduli set $\{2^n - 1, 2^n, 2^n + 1\}$. Moreover, these designs can be used in any system that has 2^n as one of its moduli. Furthermore, the proposed technique can also be used in systems with odd dynamic ranges after applying a small modification on it.

The rest of this paper is organized as follows; a brief introduction to the RNS is provided in Section II. The proposed methods for overflow detection and correction in unsigned and signed RNS are demonstrated in Sections III and IV, respectively. In Section V, the performances of the proposed designs are evaluated and compared with residue-to-binary converters and residue comparators that can be used for overflow and sign detection in both unsigned and signed RNS. Finally, the conclusion is drawn in Section VI.

II. RNS Background

The RNS is defined by a set of positive pairwise relatively prime numbers {m_1, m_2,…, m_n} called moduli. In this system, each weighted number X is uniquely represented by an ordered set of residues (x_1, x_2,…, x_n). Each residue x_i is represented by:

$$x_i = X \bmod m_i = \langle X \rangle_{m_i} \quad ; \quad 0 \le x_i < m_i \tag{1}$$

The dynamic range of this system is defined as $M = m_1 \times m_2 \times \ldots \times m_n$. Both unsigned and signed integers can be represented in RNS. For unsigned RNS, the range of representable integers is,

$$0 \le X \le M - 1 \tag{2}$$

For signed RNS, the range of representable integers is partitioned into two equal intervals,

$$\begin{array}{ll} 0 \le X < \lfloor M/2 \rfloor & \textit{for positive numbers} \\ \lfloor M/2 \rfloor \le X < M & \textit{for negative numbers} \end{array} \tag{3}$$

In this system, the arithmetic operations (addition, subtraction and multiplication) are performed totally in parallel on those very independent residues.

$$X \circ Y = \{\langle x_1 \circ y_1 \rangle, \langle x_2 \circ y_2 \rangle, \ldots, \langle x_n \circ y_n \rangle\}; \ \circ \equiv \sim (+,-,\times) \tag{4}$$

A residue number can be converted back into its binary equivalent, by using one of the residue-to-binary conversion algorithms, such as, the Chinese Reminder Theorem (CRT), the Mixed-Radix Conversion (MRC), the new CRT-I, the new CRT-II, etc. [1], [2].

According to the CRT, a residue number (x_1, x_2,…, x_n) can be converted back into its binary equivalent X by,

$$X = \left\langle \sum_{i=1}^{n} \langle x_i N_i \rangle_{m_i} M_i \right\rangle_M \tag{5}$$

where, $M_i = M / m_i$ and $N_i = \langle M_i^{-1} \rangle_{m_i}$ is the multiplicative inverse of M_i modulo m_i.

III. Proposed Method for Overflow Detection in Unsigned RNS

As aforementioned in the introduction, the general way to detect overflow is via comparing the sum with one of the addends, i.e. If $X \ge 0$ and $Y < M$ then $(X+Y)$ mod M causes overflow if and only if the sum is less than X.

Our method also depends on comparison; however, we compare each of the addends with half of the RNS range (M/2).

To detect overflow during the addition of two addends X and Y in unsigned RNS based on the moduli set {$2^n - 1$, 2^n, $2^n + 1$}, a single bit, that indicates to which half of the dynamic range M that addend belongs, is used. Based on this bit, the following three cases should be considered. The overflow will definitely occur if both of the addends are equal or greater than the half of the dynamic range M/2. No overflow will definitely occur if both of the addends are less than M/2. However, if just one of the addends is equal to or greater than M/2, then the overflow prediction becomes complex and requires further processing and evaluation of the sum (Z).

The magnitude evaluation of the addends (X and Y) is represented by a single bit (*evlt_bit*).

$$evlt\_bit_X = \begin{cases} 0 & ; \quad X < M/2 \\ 1 & ; \quad X \ge M/2 \end{cases} \tag{6}$$

The processing of the *evlt_bit* of the two addends results in the three following cases,

$$overflow = \begin{cases} 0 & ; \quad evlt\_bit_X + evlt\_bit_Y = 0 \\ 1 & ; \quad evlt\_bit_X \bullet evlt\_bit_Y = 1 \\ 'u' & ; \quad evlt\_bit_X \oplus evlt\_bit_Y = 1 \end{cases} \tag{7}$$

where, 'u' indicates the undetermined case of overflow occurrence and $(+,\bullet,\oplus)$ refer to the logical gates (OR, AND, XOR), respectively.

In order to solve the undetermined case 'u', the *evlt_bit* of the binary sum (Z) should be calculated by (6). Then the overflow can be indeed detected,

$$overflow = \begin{cases} 0 & ; \quad 'u' \ \& \ evlt\_bit_Z = 1 \\ 1 & ; \quad 'u' \ \& \ evlt\_bit_Z = 0 \end{cases} \tag{8}$$

Fig. 1 shows the structure of the proposed design that detects the overflow in unsigned RNS based on {$2^n - 1$, 2^n, $2^n + 1$}.

The magnitude evaluation of the addends and their sum, based on (6), is realized by an AND gate with (2n) inputs and an OR gate with two inputs. The magnitude evaluation unit is shown in Fig. 1 (a).

The overflow detection unit, based on (7) and (8), is realized by a 2:1 multiplexer and a XOR gate. This unit is shown in Fig. 1 (b).

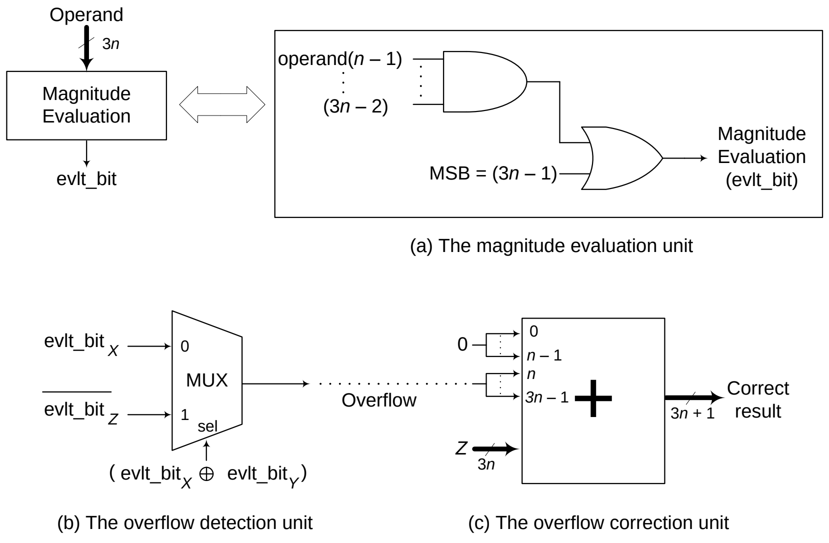

Figure 1. The internal structure of the overflow detection & correction unit for unsigned numbers : (a) Magnitude evaluation unit. (b) Overflow detection unit. (c) Overflow correction unit.

The last component of the proposed design is the overflow correction unit, which is shown in Fig. 1 (c). This unit adds back M to the sum (Z) in order to correct the overflow and obtain the final accurate result. The adder that performs the final addition can be of any type, according to the design's goal and strategy.

IV. Proposed Method for Sign and Overflow Detection in Signed RNS

In a similar manner, to detect overflow in the addition of two addends *X* and *Y* in signed RNS based on the moduli set $\{2^n - 1, 2^n, 2^n + 1\}$, a single bit, that indicates the sign of that addend, is used. As mentioned previously, in signed RNS, the positive numbers fall in the first half of the dynamic range, whereas, the negative ones fall in the second half. Thus, we have the following two cases that should be considered. No overflow will definitely occur if each of the addends has a different sign (fall in a different interval of *M*). Overflow may or may not occur if both addends have the same sign. Consequently, further processing should be done.

The sign evaluation of the addends is also represented by a single bit *evlt_bit* that is calculated by (6).

The processing of the *evlt_bit* of the two addends results in the two following cases,

$$overflow = \begin{cases} 0 & ; \ evlt\_bit_X \oplus evlt\_bit_Y = 1 \\ 'u' & ; \ evlt\_bit_X \oplus evlt\_bit_Y = 0 \end{cases} \quad (9)$$

where, '*u'* indicates the undetermined case of overflow occurrence and ⊕ refers to the logical gate XOR.

In order to solve the undetermined case '*u*', the *evlt_bit*, that determines the sign of the binary sum (*Z*) should be calculated by (6). Then the overflow can be indeed detected,

$$overflow = \begin{cases} 0 & ; \ 'u' \ \& \ evlt\_bit_Z = evlt\_bit_X \\ 1 & ; \ 'u' \ \& \ evlt\_bit_Z = \overline{evlt\_bit_X} \end{cases} \quad (10)$$

where, $\overline{evlt\_bit_X}$ refers to the logical negation of $evlt\_bit_X$.

Fig. 2 shows the structure of the proposed design that detects the sign and overflow in signed RNS based on $\{2^n - 1, 2^n, 2^n + 1\}$.

The sign evaluation of the addends and their sum, based on (6), is realized by an identical structure to that of the magnitude evaluation unit of the proposed design for unsigned RNS. It is shown in Fig. 2 (a).

The overflow detection unit, based on (9) and (10), is realized by a similar structure to that shown in Fig. 1 (b). It consists of a 2:1 multiplexer and two XOR gates. This unit is shown in Fig. 2 (b).

The overflow correction unit has an identical structure to that of the unsigned RNS. It is shown in Fig. 2 (c). Similarly, the adder that performs the final addition can be of any type.

V. Performance Evaluation and Comparison

Our proposed designs were compared with other two. The first one represents the general approach for overflow detection, which consists of a binary comparator based on the residue-to-binary converter presented in [8]. Whereas, the second one is an efficient residue comparator for general moduli set introduced in [6].

For the sake of fair comparison, unit gate model was used in order to estimate the time and area consumptions in the compared designs. According to the unit gate model, the delay (T) and area (A) of an inverter (NOT gate) were ignored.

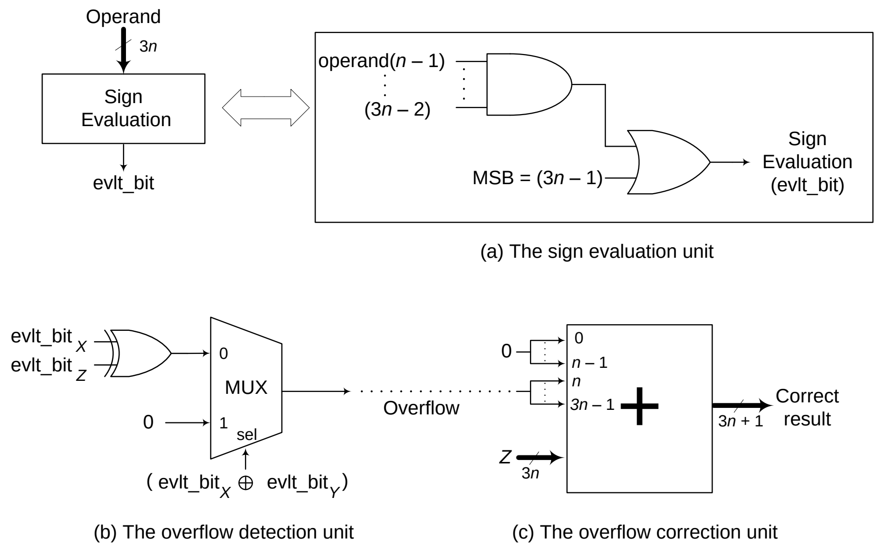

Figure 2. The internal structure of the sign and overflow detection & correction unit for signed numbers: (a) Sign evaluation unit. (b) Overflow detection unit. (c) Overflow correction unit.

Each 2-input monotonic gate (AND, NAND, OR, NOR): T = 1, A = 1. Each 2-input XOR, XNOR: T = 2, A = 2. A 2:1 multiplexer: T = 2, A = 3. Full adder: T = 4, A = 7. We have considered the 1st complement adder as a carry propagate adder with end-around carry (CPA-EAC): T= 8n, A = 7n. An n-bit binary comparator: T = 2n, A = 2n [9].

Table 1 summarizes the comparison and shows the delay and complexity of the proposed designs and the analogous ones.

The structures of the proposed designs are very similar; the only difference is an additional XOR gate in the overflow detection unit for signed RNS. Both of them are based on the residue-to-binary converter proposed in [8], the operand evaluation units of the addends and sum and the overflow detection unit. However, the evaluation of the addends is performed in parallel with the binary-to-residue conversion. Thus, no extra delay of these two units is presented. The *evlt_bit* of the addends are stored in two cells of RAM, each of them has a size of 1-bit. The correction unit was not included in the comparison. Thus, the critical path is composed of the residue-to-binary converter, the operand evaluation unit of the sum and the overflow detection unit.

The first analogous design is a binary comparator based on the reverse converter proposed in [8]. This method uses two binary comparators with a 2:1 multiplexer. The sizes of the binary comparators are 2n-bit and n-bit. Thus, the critical path in this design is composed of the residue-to-binary converter, the 2n-bit comparator and the 2:1 multiplexer.

The second analogous design presented in [6] uses a special component for generating two numbers (A_x and B_x) which are further used in the comparison. Moreover, this method uses three binary comparators and two 2:1 multiplexers. The sizes if these comparators are n-bit, n-bit and (n+1)-bit. Thus, the critical path of this circuit is composed of the A_x and B_x generator, the (n+1)-binary comparator and the two 2:1 multiplexers.

TABLE I. PERFORMANCE COMPARISON BETWEEN THE PROPOSED DESIGNS AND THE ANALOGOUS ONES

Design	Delay	Complexity
Residue comparator based on [8]	20n + 10	48n + 3
[6]	18n + 14	40n + 8
Proposed-unsigned	16n + log n + 13	37n + 18
Proposed-signed	16n + log n + 15	37n + 20

According to Table 1, both proposed designs have less delay and complexity without compromising on accuracy. Generally, lower area consumption leads to lower power consumption. However, we have not computed the estimated power consumption. Moreover, the analogous designs have not presented their power consumptions.

In case of overflow occurrence, M is added back to the binary sum, in order to correct the sum (Z) and get the final accurate result. The adder that performs this addition can be of any type, based on the design's goal and strategy. Moreover, the size of this adder is 2n-bit instead of 3n-bit, since the first n-bits of M for the moduli set {2^n – 1, 2^n, 2^n + 1} are '0'.

$$\begin{aligned} overflow = \text{'0'} &\Rightarrow \quad final\ result = Z + \text{"}\underbrace{00\cdots00}_{2n}\underbrace{0\cdots0}_{n}\text{"} \\ overflow = \text{'1'} &\Rightarrow \quad final\ result = Z + \text{"}\underbrace{11\cdots11}_{2n}\underbrace{0\cdots0}_{n}\text{"} \end{aligned} \tag{11}$$

We have mentioned that both proposed designs can be used with any RNS that uses any moduli set that has (2^n) as one of its moduli, i.e. has an even dynamic range. This can be simply performed by applying tiny modification on the evaluation units, represented in changing the number of the inputs of the AND gate, according to the dynamic range provided by the used moduli set. Moreover, in case of changing the number and the order of the AND gate's inputs, these two designs can be used with any other moduli set, even if it provides an odd dynamic range. However, for such systems, the parity checking technique will be faster and simpler than the proposed one.

VI. CONCLUSIONS AND FUTURE WORK

Overflow detection is one of the main challenges in the RNS, especially in systems based on moduli sets that provide even dynamic ranges. This paper presented two designs for overflow and sign detection and correction in unsigned and signed RNS based on the moduli set $\{2^n - 1, 2^n, 2^n + 1\}$. This set has an even dynamic range. Moreover, these designs can be considered as universal, since they can be used with any system that has an even dynamic range by applying a small modification on the evaluation unit. Both designs are faster and require less hardware components than those based on comparators. Our next step will be dedicated on designing a FIR filter based on RNS that uses the proposed sign and overflow detection and correction units in order to highlight their efficiency when embedded in a DSP application.

ACKNOWLEDGMENT

This research was supported by the Ministry of Industry and Trade of the Czech Republic under the MPO ČR č. FR-TI3/485 project and Prospective applications of new sensor technologies and circuits for processing of sensor signals, No.FEKT-S-11-16 project.

REFERENCES

[1] A. Omondi and B. Premkumar, Residue Number Systems: Theory and Implementation. Imperial College Press, UK, 2007.

[2] M. Lu, Arithmetic and Logic in Computer Systems. John Wiley & Sons, Inc., New Jersey, USA, 2004.

[3] M. Askarzadeh, M. Hosseinzadeh and K. Navi, "A new approach to overflow detection in moduli set $\{2^n - 3, 2^n - 1, 2^n + 1, 2^n + 3\}$," Second International Conference on Computer and Electrical Engineering (ICCEE'09), 28-30 December 2009, pp. 440 – 441.

[4] M. Shang, H. JianHao, Z. Lin and L. Xiang, "An efficient RNS parity checker for moduli set $\{2^n - 1, 2^n + 1, 2^{2n} + 1\}$ and its applications," Springer Journal of Science in China Series F: Information Sciences, 10 November 2008, pp. 1567 – 1570.

[5] B. Zarei, M. Askarzadeh, N. Derakhshanfard and M. Hosseinzadeh, "A high-speed residue number comparator for the 3-moduli set $\{2^n-1, 2^n+1, 2^n+3\}$," International Signals Systems and Electronics (ISSSE), 17-20 September 2010, pp. 2 – 3.

[6] S. Bi and W. J. Gross, "Efficient residue comparison algorithm for general moduli sets," 48th Midwest Symposium on Circuits and Systems, 7-10 August 2005, pp. 1602 – 1604.

[7] G. Dimauro, S. Impedovo and G. Pirlo, "A new technique for fast number comparison in the residue number system," IEEE Transactions on Computers, May 1993, pp. 608 – 611.

[8] S. J. Piestrak, "A high-speed realization of a residue to binary number system converter," IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, October 1995, pp. 661 – 663.

[9] F. Vahid, Digital Design with RTL Design, VHDL and Verilog, John Wiley & Sons Publishers, USA, 2011.