



International Journal of Emerging Technologies in Computational and Applied Sciences (IJETCAS)

(Open Access, Double Blind Peer-reviewed, Refereed and Indexed Journal)

www.iasir.net

The Chinese Remainder Theorem Method for Detecting and Correcting Overflow in Residue Number System Addition for the Moduli Set $\{2^{2n}-1, 2^{2n}, 2^{2n}+1\}$

E. K. Amegan¹, A. Alhassan¹ and S. Akobre¹

¹Department of Computer Science, Faculty of Mathematical Sciences,
University for Development Studies, Ghana.

Abstract: This paper focuses on efficient overflow detection and correction algorithm in Residue Number System (RNS) architecture. A generalized algorithm is first proposed and then implemented on the moduli set $\{2^{2n}-1, 2^{2n}, 2^{2n}+1\}$. The Chinese Remainder Theorem (CRT) is employed to formulate the proposed algorithm. The proposed scheme detects and corrects overflow during RNS summation of two numbers X and Y. The partial reverse conversion technique is used in the overflow detection process that gives accurate results. The delay and area requirement of the proposed scheme is compared with other existing state-of-the-art schemes. Which shows better performance in terms of size and speed requirements.

Keywords: Residue Number System, dynamic range, Chinese Remainder Theorem, Greatest Common Divisor, Overflow.

I. INTRODUCTION

In digital computing, the study of number systems is very necessary as computers generally work on numbers. One important number system today that has seen little attention is the Residue Number System (RNS). Basically what RNS does is to simplify the computation of large numbers by decomposing each of these numbers into a set of residues generated with respect to a particular moduli set. Then parallel computation is carried out with respect to these residues. Encoding a large number into a group of small numbers results in significant speed up of the overall data processing. These features enable RNS utilization in the fields of Digital Signal Processing (DSP), intensive computations like digital filtering, convolutions, correlations, Discrete Fourier Transform (DFT) computations, Fast Fourier Transform (FFT) computations and direct digital frequency synthesis [1], [6], [7]. However, RNS is not very popular in processor construction due to some difficult arithmetic operations such as overflow detection, magnitude comparison, and reverse conversion amongst others. However, solutions to some of these problems are currently being developed as a result of ongoing research. Unlike Weighted Number System (WNS), where overflow can be efficiently handled by rounding, truncating or saturating arithmetic, overflow detection in RNS involves more complex and time consuming procedures.

Given a set of pair-wise relatively prime numbers $\{m_1, m_2, \dots, m_n\}$ with a residue representation $\{r_1, r_2, \dots, r_n\}$ in RNS of some integer number X, i.e $r_i = |X|_{m_i}$, that number and its residues are related by the equation:

$$X = \left| \sum_{i=1}^n r_i |M_i^{-1}|_{m_i} M_i \right|_M \quad (1)$$

Where: M is the product of the m_i 's and $M_i = \frac{M}{m_i}$. In

RNS, Overflow is defined to be a condition where a number which falls outside the legitimate range of a particular RNS i.e. $[0, M-1]$ is well represented as a legitimate RNS number [5]. For example, the sum of decimal numbers 50 and 15 is 65. Performing this addition in RNS using the moduli set $\{3, 4, 5\}$ generates $(2, 1, 0)_{\text{RNS}\langle 3|4|5 \rangle}$ as results. $(2, 1, 0)_{\text{RNS}\langle 3|4|5 \rangle}$ is the equivalent of decimal number 5. This is because, the sum of 50 and 15 which is 65 falls outside the legitimate range, hence there is an overflow in the sum. The traditional overflow detection technique utilizes either the Chinese Remainder Theorem (CRT) [5] or the Mixed Radix Conversion (MRC) techniques.

Over the years, many researchers have made considerable efforts to design overflow detection schemes which do not require full reverse conversion. Previously proposed RNS overflow detection algorithms still rely on operands examination by [5] and other costly and time consuming procedures such as base extension, use of redundant RNS, group number approach and sign detections as in [9] and [8]. The scheme in [5], is demonstrated to be better than those in [3] and [9] in terms of both area and delay.

This paper presents a scheme for RNS overflow detection and correction. The scheme uses partial reverse conversion based on the CRT technique. The rest of the paper is organized as follows. In Section 2, the method is proposed. Section 3, comprises of the hardware implementation of the scheme with a simplified algorithm and numerical examples. The performance of the scheme is evaluated in Section 4 while the paper is concluded in Section 5.

II. PROPOSED METHOD

The proposed method using Chinese Remainder Theorem (CRT) is presented in detail in this section.

The algorithm for the proposed method is as follows: [10]

1. Accept X and Y
2. Determine α_x and α_y according to (7)
3. Determine E and β according to (10) and (11)
4. Overflow occurs only under one of the following conditions:
 - (i) If the MSB of E i.e. $E_{4n} = 1$
 - (ii) If E_{4n-1} down to E_0 is "1"
 - (iii) If E_{4n-1} down to E_1 is "1" and $\beta = 1$
5. The correct result is computing Z according to (10)

Given the RNS number $X = (x_1, x_2, x_3)$ and $Y = (y_1, y_2, y_3)$ with respect to the moduli set $\{2^{2n}-1, 2^{2n}, 2^{2n}+1\}$, where $m_1 = 2^{2n}-1$, $m_2 = 2^{2n}$ and $m_3 = 2^{2n}+1$. Then,

$$M_1 = 2^{2n}(2^{2n}+1), M_2 = (2^{4n}-1), \quad M_3 = 2^{2n}(2^{2n}-1) \quad (2)$$

Lemma 1 :

For the given moduli set, the respective inverses are;

$$|M_1^{-1}|_{m_1} = 2^{2n-1} \quad (3)$$

$$|M_2^{-1}|_{m_2} = -1 \quad (4)$$

$$|M_3^{-1}|_{m_3} = -2^{2n-1} \quad (5)$$

The proof of (3) – (5) is demonstrated in [11]

Lemma 2:

Using CRT in (1), the binary number can be written as:

$$X = |2^{2n}(2^{2n}+1) \times (2^{2n-1})x_1 + (2^{4n}-1)(-1)x_2 + 2^{2n}(2^{2n}-1)(-2^{2n-1})x_3|_{2^{4n}(2^{4n}-1)}$$

Subtracting x_2 from both sides of the above expression and dividing by 2^{2n} , the binary number is obtained as:

$$X = 2^{2n}\alpha + x_2 \quad (6)$$

Where

$$\alpha = \left| \frac{(2^{2n}+1)(2^{2n-1})x_1 - 2^{2n}x_2 - (2^{2n}-1)(2^{2n-1})x_3}{2^{4n}-1} \right|_{2^{4n}-1} \quad (7)$$

From (6), let X and Y be two RNS numbers such that their sum is Z.

From (6) it implies that:

$$X = 2^{2n}\alpha_x + x_2 \quad (8)$$

$$Y = 2^{2n}\alpha_y + y_2 \quad (9)$$

$$Z = 2^{2n}(\alpha_x + \alpha_y) + (x_2 + y_2), Z = 2^{2n}E + R \quad (10)$$

Where $E = (\alpha_x + \alpha_y)$ and $R = (x_2 + y_2)$

Let

$$\beta = \begin{cases} R < 2^{2n}, 0 \\ R \geq 2^{2n}, 1 \end{cases} \quad (11)$$

Lemma 3:

Given any two (2) RNS numbers $X = (x_1, x_2, x_3)$ and $Y = (y_1, y_2, y_3)$, overflow occurs if and only if;

$$E \geq 2^{4n}-1 \quad (12)$$

or

$$E = 2^{4n}-2 \text{ and } \beta = 1 \quad (13)$$

Proof: Assume (12) holds true; then for (10)

$$Z \geq 2^{2n}(2^{4n}-1) + R$$

$$Z \geq M + R$$

Which is outside the legitimate range, i.e. $[0, M-1]$, hence overflow will occur.

Furthermore, if (13) holds true, then (10) can be rewritten as;

$$Z = 2^{2n}(2^{4n}-2+1)$$

$$Z = 2^{2n}(2^{4n}-1)$$

$$Z = M$$

Which is also outside the legitimate range, therefore overflow will occur. Hence the proof.

From equation (10), Z will be the correct result of summing X and Y whether overflow occurs or not in the given moduli set, but will be out of the range in $[0, M-1]$ if either (12) or (13) holds; therefore, E should be added to the Decimal to Residue (DR) to be $[0, M+E-1]$ in order to legitimize Z.

III. HARDWARE IMPLEMENTATION

In order to reduce the hardware complexity, we use the following properties to simplify equation (7).

Property 1: The multiplication of a residue number v by 2^P in modulo $(2^n - 1)$ is carried out by P bit circular left shift, where P is a natural number.

Property 2: The residue of a negative residue number $(-v)$ in modulo $(2^n - 1)$ is the one's complement of v , where $0 \leq v < 2^n - 1$.

Equation (7) can further be simplified as follows:

$$\alpha = |(2^{2n}x_1 + x_1)2^{2n-1} - 2^{2n}x_2 - (2^{2n}x_3 - x_3)2^{2n-1}|_{2^{4n-1}} \quad (14)$$

$$\text{Let } \psi_1 = |(2^{2n}x_1 + x_1)2^{2n-1}|_{2^{4n-1}} \quad (15)$$

$$\psi_2 = |-2^{2n}x_2|_{2^{4n-1}} \quad (16)$$

$$\psi_3 = |-2^{4n-1}x_3|_{2^{4n-1}} \quad (17)$$

$$\psi_4 = |2^{2n-1}x_3|_{2^{4n-1}} \quad (18)$$

It is necessary to note that $x_{i,j}$ means the j^{th} bit of x_i .

Evaluation of ψ_1

The residue x_1 can be represented as follows: (19)

$$\begin{aligned} \psi_1 &= |(2^{2n}x_1 + x_1)2^{2n-1}|_{2^{4n-1}} = \left| 2^{2n-1} \left(\underbrace{x_{1,2n-1} \dots x_{1,1}}_{4n\text{-bits}} \underbrace{x_{1,0}}_{2n\text{ Bits}} + \underbrace{0 \dots 0}_{2n\text{ Bits}} \underbrace{x_{1,2n-1} \dots x_{1,0}}_{4n\text{-bits}} \right) \right|_{2^{4n-1}} \\ &= \left| 2^{2n-1} \left(\underbrace{x_{1,2n-1} \dots x_{1,1} x_{1,0}}_{2n+1} \underbrace{x_{1,2n-1} \dots x_{1,1} x_{1,0}}_{4n\text{ bits}} \right) \right|_{2^{4n-1}} \\ &= \underbrace{x_{1,0} x_{1,2n-1} \dots x_{1,1} x_{1,0}}_{2n+1} \underbrace{x_{1,2n-1} \dots x_{1,1}}_{2n-1} \end{aligned} \quad (20)$$

Evaluation of ψ_2

The residue x_2 can be represented as follows;

$$x_2 = x_{2,2n-1} \dots x_{2,1} x_{2,0} \quad (21)$$

Therefore;

$$\begin{aligned} \psi_2 &= |-2^{2n}x_2|_{2^{4n-1}} \\ &= \bar{x}_{2,n-1} \dots \bar{x}_{2,1} \bar{x}_{2,0} \underbrace{11 \dots 11}_{2n\text{ bits}} \end{aligned} \quad (22)$$

Where $\bar{x}$ means the complement of x

Evaluation of ψ_3 and ψ_4

The residue x_3 can be represented as follows:

$$x_{3,2n} \dots x_{3,1} x_{3,0} \quad (23)$$

Therefore

$$\begin{aligned} \psi_3 &= |-2^{4n-1}x_3|_{2^{4n-1}} \\ &= \bar{x}_{3,0} \underbrace{11 \dots 11}_{2n-1} \underbrace{\bar{x}_{3,2n} \bar{x}_{3,2n-1} \dots \bar{x}_{3,1}}_{2n} \end{aligned} \quad (24)$$

$$\begin{aligned} \psi_4 &= |2^{2n-1}x_3|_{2^{4n-1}} \\ \psi_4 &= \underbrace{x_{3,2n} \dots x_{3,1} x_{3,0}}_{2n+1} \underbrace{00 \dots 00}_{2n-1} \end{aligned} \quad (25)$$

Correction Unit

In order to evaluate the sum Z , we further simplify equation (10)

$$Z = \tau + R \quad (26)$$

$$\tau = 2^{2n}E$$

$$= \underbrace{E_{4n} E_{4n-1} \dots E_1 E_0}_{6n+1\text{ bits}} \underbrace{00 \dots 0}_{2n\text{ bits}} \quad (27)$$

$$R = \underbrace{R_{2n} R_{2n-1} \dots R_1 R_0}_{2n+1\text{ Bits}} \quad (28)$$

$$\text{Therefore, } = \underbrace{\tau_{6n} \tau_{6n-1} \dots \tau_1 \tau_0}_{6n+1\text{ bits}} + \underbrace{00 \dots 0 R_{2n} R_{2n-1} \dots R_1 R_0}_{4n\text{ bits}} \quad (29)$$

Implementation of equations (26) - (29) gives the correct result of Z whether overflow occurs or not.

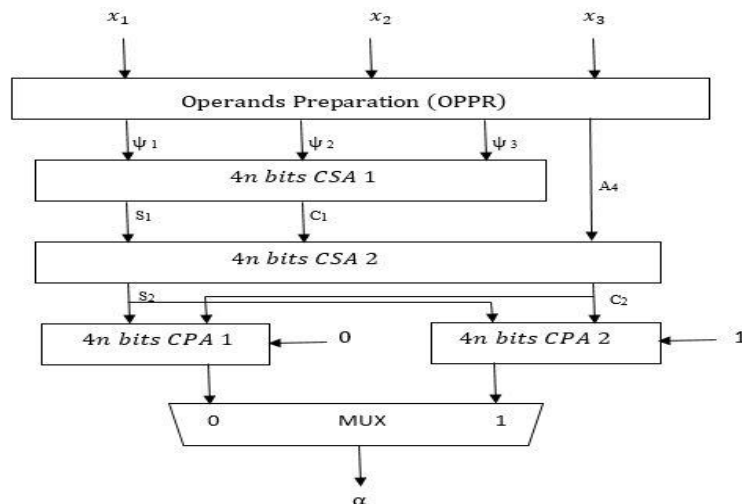


Figure 1: Block diagram of the partial reverse converter

A. Proposed Architecture

The parameter α which is the output of the partial reverse converter is computed according to equation (14) where all the parameters are defined in equations (15) -(18). For two numbers X and Y , α_x and α_y are the α values corresponding to X and Y respectively and are computed using Carry Save Adders (CSAs) 1 and 2 and two regular $4n$ -bits Carry Propagate Adders (CPAs) 1 and 2. The results of these CPAs are passed on to a multiplexer (MUX 1) which will then pass either of them down. MUX 1 will pass on the result of CPA 1 if the carry out of CSA 1 is a '0', otherwise the result of CPA 2 is passed on.

α_x , corresponding to the decimal number X and α_y , corresponding to the decimal number Y is added using a regular $(4n + 1)$ -bits CPA 3 in order to get E ; at the same time, x_2 and y_2 is computed using a regular $(2n + 1)$ -bits CPA 4 to obtain R .

A multiplexer (MUX 2) is used to select the value of α to be zero if the most significant bit (MSB) of R is 0, otherwise, it selects one (1) if the MSB of R is 1. This is shown in Figure 2 which is the overflow detection unit. The area required by CSAs 1 and 2 is $4nA_{FA}$ each. CPAs 1 and 2 require $4nA_{FA}$ as well. Therefore, a total area of $16nA_{FA}$ is required to obtain α . So for two numbers X and Y , the total area requirement will be $32nA_{FA}$. CPA 3 demands an area of $(4n + 1) A_{FA}$ and CPA 4 also requires $(2n + 1) A_{FA}$ of resources. Thus, the area requirement for the overflow detection component is $(6n + 2) A_{FA}$. Therefore, the total area requirement of the overflow detection scheme is $(38n + 2) A_{FA}$. Regarding the delay, each CSA (i.e. CSAs 1 and 2) impose a delay of D_{FA} while each CPA (1 and 2) imposes a delay of $4nD_{FA}$. Since they are in parallel, for two numbers this will become $8nD_{FA}$, thus delay imposed on computing α is $(8n + 2)D_{FA}$. Also the CPA pair 3 and 4 impose a delay of $(4n + 1) D_{FA}$ for the overflow detection unit. Therefore, the delay required for the proposed scheme is $(12n + 3) D_{FA}$. The correction unit uses a regular $(6n + 1) bits$ CPA 5. The area requirement is $(6n + 1) A_{FA}$ and its delay is also $(6n + 1) D_{FA}$.

Schematic Diagrams of the Proposed Schemes:

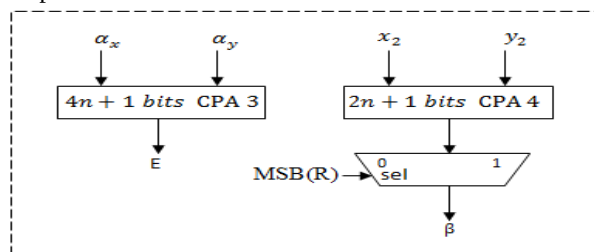


Figure 2: Overflow Detection Unit

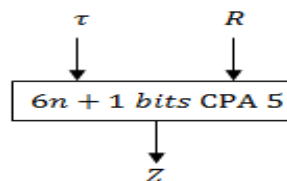


Figure 3: Correction Unit

B. Numerical Illustrations

In this section, some numerical examples with the proposed scheme is presented.

For the moduli set: $\{2^{2n}-1, 2^{2n}, 2^{2n}+1\}$, when $n = 2$, then the moduli set is; $\{15, 16, 17\}$.

- a. Given two (2) numbers $X = 1510$ and $Y = 500$, overflow is checked for in the sum of X and Y using moduli set $\{15, 16, 17\}$.

Thus, $1510 = (10,6,14)_{RNS\{15|16|17\}}$

$$= (1010, 0110, 01110)_{RNS\{1111|10000|10001\}}$$

$$500 = (5, 4, 7)_{RNS\{15|16|17\}}$$

$$= (0101, 0100, 00111)_{RNS\{1111|10000|10001\}}$$

Therefore;

$$1510 + 500 = ((1010, 0110, 01110) + (0101, 0100, 00111))_{RNS\{1111|10000|10001\}}$$

$$= (0000, 1010, 0100)_{RNS\{1111|10000|10001\}}$$

RNS to decimal conversion of $(0000, 1010, 0100)_{RNS\{1111|10000|10001\}}$ will result in the decimal number 2010 which is the correct result of $1510 + 500$

Checking for RNS overflow using the proposed algorithm;

$$\theta_x = 94 = 001011110$$

$$\theta_y = 31 = 000011111$$

$$E = 001011110 + 000011111 = 001111101$$

$$R = 0110 + 0100 = 1010, \alpha = 0$$

After processing, the scheme will obviously detect no overflow because none of the conditions are met.

- b. Given two (2) numbers $X = 2040$ and $Y = 3010$, we then check for overflow in the sum of X and Y using moduli set $\{15, 16, 17\}$. Thus

$$2040 = (0,8,0)_{RNS\{15|16|17\}}$$

$$= (0000, 1000, 0000)_{RNS\{1111|10000|10001\}}$$

$$3010 = (10, 2, 1)_{RNS\{15|16|17\}}$$

$$= (1010, 0010, 00001)_{RNS\{1111|10000|10001\}}$$

Therefore;

$$2040 + 3010 = ((0000, 1000, 00000) + (1010, 0010, 00001))_{RNS\{1111|10000|10001\}}$$

$$= (1010, 1010, 00001)_{RNS\{1111|10000|10001\}}$$

RNS to decimal conversion of

$$(1010, 1010, 00001)_{RNS\{1111|10000|10001\}}$$

will result in the decimal number 970. While the sum of the decimal numbers 2040 and 3010 is 5050. Therefore, a sign that overflow has occurred.

Checking for RNS overflow using the proposed algorithm

$$\theta_x = 127 = 001111111$$

$$\theta_y = 188 = 010111100$$

$$E = 001111111 + 010111100 = 100111011$$

$$R = 1000 + 0010 = 1010, \alpha = 0$$

Since the MSB of E is "1", the scheme will detect that overflow has occurred.

IV. PERFORMANCE EVALUATION

In this section, the performance results of the proposed scheme is compared with [3]. Theoretical analysis from Table 1 shows that as n gets bigger, the proposed scheme performs faster than the scheme in [3]. In terms of delay, the proposed scheme requires lesser resources. From the AD^2 comparison, it can be concluded that the proposed scheme is efficient.

Table 1. Area, Delay, AD^2 Comparison

SCHEME	[3]			PROPOSED		
n	Delay ($16n + \log n + 13$)	Area ($37n + 18$)	AD^2 ($9472n^3 + 20000n^2 + 13661n + 3042$)	Delay ($12n + 3$)	Area ($38n + 2$)	AD^2 ($5472n^3 + 3024n^2 + 486n + 18$)
2	22.63	92	186140	27	78	56862
4	51.53	166	983894	51	154	400554
6	87.70	240	2850960	75	230	1293750
8	128.60	314	6241994	99	306	2999106
10	173.00	388	11611652	123	382	5779278

12	220.20	462	19414590	147	458	9896922
14	269.73	536	30105464	171	534	15614694
16	321.25	610	44138930	195	610	23195250
18	374.52	684	61969644	219	686	32901246
20	429.33	758	84052262	243	762	44995338

V. CONCLUSION

In this research, RNS overflow detection and correction during addition is discussed. The problem of overflow has prevented the use of RNS in general purpose computing. However, for successful application of RNS, overflow detection must be easy and fast. The paper presented a general algorithm in detecting overflow on the moduli set $(2^{2n}-1, 2^{2n}, 2^{2n}+1)$. The approach does not require full RNS-binary conversion. Also the approach gives the correct result whether overflow occurs or not during the addition of two numbers. The proposed scheme is demonstrated to be more efficient.

REFERENCES

- [1] A. Omondi and B. Premkumar (2007). Residue Number Systems Theory and Implementation (Vol 2). Imperial College Press.
- [2] A. S. Molahosseini, K. Navi (2007). New Arithmetic residue to binary Converters. International Journal of Computer Sciences and Engineering Systems, Vol. 1, No.4, pp. 295-299
- [3] D. Younes and P. Steffan (2013). Universal approaches for overflow and sign detection in residue number system based on $\{2^n-1, 2^n, 2^n+1\}$. The Eighth International Conference on Systems (ICONS 2013), pp. 77 – 84.
- [4] E. K. Bankas and K. A. Gbolagade (2013). A New Efficient FPGA Design of Residue-to-binary Converter. International Journal of VLSI design & Communication Systems (VLSICS) Vol.4, No.6.
- [5] H. Siewobr and K. A. Gbolagade (2014). RNS Overflow Detection by Operands Examination. International Journal of Computer Applications (0975 – 8887), Vol 85, No. 18 .
- [6] K. A. Gbolagade (2013) . An Efficient MRC based RNS-to Binary Converter for the $\{2^{2n}-1, 2^n, 2^{2n+1}-1\}$ Moduli Set. International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 2, Issue 4.
- [7] K.A. Gbolagade, R. Chaves, L. Sousa, and S.D. Cotozana (2010). An improved reverse converter for the $\{2^{2n+1}-1, 2^n, 2^n-1\}$ moduli set. IEEE International Symposium on Circuits and Systems (ISCAS 2010), pp. 2103-2106.
- [8] L. Theodore Houk (1989). Residue Addition Overflow Detection Processor. Boing Company, Seattle, Wash. Appl. No.:414276.
- [9] M. Rouhifar, M. Hosseinzadeh, S. Bahanfar and M. Teshnehlab (2011). Fast Overflow Detection in Moduli Set. International Journal of Computer Science Issues, Vol. (8/3), pp. 407-414.
- [10] P. A. Agbedemab and E.K. Bankas (2015). A Novel RNS Overflow Detection and Correction Algorithm for the Moduli Set $\{2^n-1, 2^n, 2^n+1\}$. International Journal of Computer Applications (0975 – 8887)
- [11] P. A. Mohan (2007). Reverse Converters for a New Moduli Set $\{2^{2n}-1, 2^n, 2^{2n}+1\}$. Circuit Systems Signal Processing, 26 : 218.