American International Journal of Research in Science, Technology, Engineering & Mathematics

Available online at http://www.iasir.net

ISSN (Print): 2328-3491, ISSN (Online): 2328-3580, ISSN (CD-ROM): 2328-3629

AIJRSTEM is a refereed, indexed, peer-reviewed, multidisciplinary and open access journal published by International Association of Scientific Innovation and Research (IASIR), USA
(An Association Unifying the Sciences, Engineering, and Applied Research)

DETECTING AND CORRECTING OVERFLOW IN RNS ADDITION BY PARTIAL REVERSE CONVERSION FOR THE MODULI SET $\{2^{2n}-1, 2^n, 2^{2n}+1\}$

M. I. Daabo[1], E. K. Amegan[2], Abdul-Mumin Salifu[3]
[1,2,3]Department of Computer Science, Faculty of Mathematical Sciences,
University for Development Studies, Ghana

Abstract: *This paper presents an efficient algorithm for detecting and correcting overflow in Residue Number System (RNS) architecture. Firstly, a generalized algorithm for overflow detection and correction is presented and then the scheme is applied on the moduli set $\{2^{2n}-1, 2^n, 2^{2n}+1\}$. The proposed algorithm is based on the Chinese Remainder Theorem and detects overflow in RNS addition of two operands. In addition, the scheme detects overflow using partial reverse conversion and gives results without errors. In order to prove the efficiency of the proposed scheme, a comparison in terms of delay and area requirements is also presented. Generally the scheme is seen to be better in terms of size and speed as compared to the best known schemes.*

General Terms: *Residue Number System, Discrete Fourier Transform, reverse converter,digital circuits, VLSI design.*

Keywords: *Residue Number System, parallel computation, Chinese Reminder Theorem, Overflow, One's complement*

I. INTRODUCTION

RNS is a form of parallel computing and an integer number system that utilizes remainders to represent numbers. It supports parallelism, carry-free addition, borrow-free subtraction and single step multiplication without partial products. These inherent features enable RNS utilization in the fields of Digital Signal Processing (DSP), computational intensive areas such as; digital filtering, convolutions, correlations, Discrete Fourier Transform (DFT) computations, Fast Fourier Transform (FFT) computations and direct digital frequency synthesis [5], [6]. However, for a successful application of RNS, overflow detection and correction must be easier and faster to perform so as not to limit the full usage of RNS in general purpose computing. In Weighted Number System (WNS), overflow can be efficiently handled by rounding, truncating or saturating arithmetic. Overflow detection in RNS involves more complex and time consuming procedures.

RNS is determined by a set S, of N integers that are pair-wise relatively prime. That is $S = \{m_1, m_2 \ldots m_N\}$ where $\gcd(m_i, m_j) = 1$ for i, j=1... N and $i \neq j$, and gcd means the greatest common divisor[1]. Every integer X in [0,−1] can be uniquely represented with N-tuple where, $M = \prod_{i=1}^{N} m_i$ $X \rightarrow (x_1, x_2, \ldots, x_N)$ and $x_i = |X|_{m_i} = (X \bmod m_i)$; for i=1 *to* N. The set S and the number x_i are called the moduli set and residue of X modulo m_i respectively. In order to calculate the number X from its residues, we can apply the CRT which relates X and its RNS representation by:

$$X = \left|\sum_{i=1}^{N} w_i x_i\right|_M \quad \textbf{(1)}$$

Where $M = \prod_{i=1}^{N} m_i$; $w_i = M_i \left|M_i^{-1}\right|_{m_i}$ with $M_i = \frac{M}{m_i}$ and M_i^{-1} being the multiplicative inverse of M_i.

The phenomenon of overflow in computing is about storing data which is more than its intended memory location. In RNS, the concept of overflow is defined to be a condition where a number falls outside the legitimate range of a particular RNS. A number from the set [0. M-l] is well represented as a legitimate RNS number [4]. For instance, let us consider the sum of the decimal numbers 90 and 25 which is 115. Performing this addition in RNS using the moduli set {3, 5, 7} with dynamic range of 105 generates $(1,0,3)_{RNS<3|5|7>}$ as the result. But the number $(1,0,3)_{RNS<3|5|7>}$ is the equivalent of the decimal number 10. This is because the sum of 90 and 25 which is 115 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) [4] or the Mixed Radix Conversion (MRC) techniques.

In recent research, researchers have made considerable efforts to design efficient overflow detection schemes which are dependent on full reverse conversion. Some proposed RNS overflow detection algorithms are based on operands examination [4] and other costly and time consuming procedures such as base extension, use of redundant RNS, group number approach and sign detections as in [8] and [7] The scheme in [4] is demonstrated to be better than those in [2] and [8] in terms of both area and delay.

In this paper, a scheme for RNS overflow detection and correction is proposed. 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. In Section 3, the hardware implementation of the scheme with a simplified algorithm and numerical examples are also presented. The performance of the scheme is evaluated in Section 4 whiles the paper is concluded in Section 5.

II. PROPOSED METHOD

The proposed algorithm using CRT is presented in details in this section. The method detects overflow and corrects it.

A. Algorithm for the Proposed Scheme

The algorithm for the proposed method is as follows: [9]

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 β= 1
5. The correct result is computing Z according to (10)

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

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

Lemma 1 :

For the given moduli set, we have

$$\left|M_1^{-1}\right|_{m_1} = 2^{n-1} \tag{3}$$

$$\left|M_2^{-1}\right|_{m_2} = -1 \tag{4}$$

$$\left|M_3^{-1}\right|_{m_3} = 2^{n-1} \tag{5}$$

The proof of (3) – (5) is demonstrated in [10]

Lemma 2:

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

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

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

$$X = 2^n\alpha + x_2 \tag{6}$$

Where

$$\alpha = \left|\begin{matrix}(2^{2n}+1)(2^{n-1})x_1 - 2^{3n}x_2 + \\ (2^{2n}-1)(2^{n-1})x_3\end{matrix}\right|_{2^{4n}-1} \tag{7}$$

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

From **(6)** it implies that:

$$X = 2^n\alpha_x + x_2 \tag{8}$$

$$Y = 2^n\alpha_y + y_2 \tag{9}$$

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

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

Let

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

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

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

or

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

Proof

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

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

$$Z \geq M + R$$

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

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

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

$$Z = M$$

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 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:

$$\alpha = \left|\left|(2^{3n-1} + 2^{n-1})x_1\right|_{2^{4n}-1} + \left|-2^{3n}x_2\right|_{2^{4n}-1} + \left|2^{3n-1}x_3\right|_{2^{4n}-1} + \left|-2^{n-1}x_3\right|_{2^{4n}-1}\right|_{2^{4n}-1} \quad \textbf{(14)}$$

Let

$$\psi_1 = \left|(2^{3n-1} + 2^{n-1})x_1\right|_{2^{4n}-1} \quad \textbf{(15)}$$

$$\psi_2 = \left|-2^{3n}x_2\right|_{2^{4n}-1} \quad \textbf{(16)}$$

$$\psi_3 = \left|2^{3n-1}x_3\right|_{2^{4n}-1} \quad \textbf{(17)}$$

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

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

Evaluation of ψ_1

The residue x_i can be represented as follows: $x_{1,2n-1} \cdots x_{1,1}\, x_{1,0}$ **(19)**

$$\psi_1 = \left|(2^{3n-1} + 2^{n-1})x_1\right|_{2^{4n}-1}$$

The residue x_1 can be represented in 4n bits as follows:

$$x_1 = \overbrace{00\ldots00}^{2n\text{ bits}}\, x_{1,2n-1} \cdots x_{1,1}x_{1,0}$$

We evaluate the two parts of ψ_1 separately using property 1

$$\left|2^{3n-1}x_3\right|_{2^{4n}-1} = \underbrace{r_{2,n} \cdots r_{2,1}r_{2,0}}_{n+1\text{ bits}}\ \overbrace{00\ldots00}^{2n\text{ bits}}\ \underbrace{r_{2,(2n-1)} \cdots r_{2,(n-1)}}_{n-1\text{ bits}}$$

$$\left|2^{n-1}x_1\right|_{2^{4n}-1} = \underbrace{00\ldots00}_{n+1\text{ bits}}\overbrace{r_{2,(2n-1)} \cdots r_{2,1}r_{2,0}}^{2n\text{ bits}}\underbrace{00\ldots00}_{n-1\text{ bits}}$$

By adding the two above expressions we have the final value of ψ_1 as

$$\psi_1 = \underbrace{r_{2,n} \cdots r_{2,1}r_{2,0}}_{n+1\text{ bits}}\ \overbrace{r_{2,(2n-1)} \cdots r_{2,1}r_{2,0}}^{2n\text{ bits}}\ \underbrace{r_{2,(2n-1)} \cdots r_{2,(n-1)}}_{n-1\text{ bits}}\ (20)$$ which is a 4n bits residue number. **(20)**

Evaluation of ψ_2

The residue x_i can be represented as follows; **(21)**

$$x_{2,n-1} \cdots x_{2,1}\, x_{2,0}$$

$$\psi_2 = \left|-2^{3n}x_2\right|_{2^{4n}-1}$$

The residue x_1 can be represented in 4n bits as follows;

$$x_2 = \overbrace{00\ldots00}^{3n\text{ bits}}\, x_{2,n-1} \cdots x_{2,1}x_{2,0}$$

By applying property 1:

$$|2^{3n}x_2|_{2^{4n}-1} = x_{2,n-1} \cdots x_{2,1}x_{2,0}\overbrace{00\ldots00}^{3n\ bits}$$

And finally by applying property 2 we get:

$$\psi_2 = |-2^{3n}x_2|_{2^{4n}-1}$$

$$= \bar{x}_{2,n-1} \cdots \bar{x}_{2,1}\bar{x}_{2,0}\overbrace{11\ldots11}^{3n\ bits} \quad \textbf{(22)}$$

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

Evaluation of ψ_3 and ψ_4

The residue x_3 can be represented as follows;

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

$$\psi_3 = |2^{3n-1}x_3|_{2^{4n}-1}$$

The residue x_3 can be represented in 4 bits as follows:

$$x_3 = \overbrace{00\ldots00}^{2n-1\ bits}\, x_{3,2n} \cdots x_{3,1}x_{3,0}$$

By applying property 1:

$$|2^{3n-1}x_3|_{2^{4n}-1} = \underbrace{x_{3,n} \cdots x_{3,1}x_{3,0}}_{n+1\ bits}\overbrace{00\ldots00}^{2n-1\ bits}\underbrace{r_{3,2n} \cdots r_{3,(n+1)}}_{n\ bits} \quad \textbf{(24)}$$

Again, $|2^{n-1}x_3|_{2^{4n}-1} = \underbrace{00\ldots00}_{n\ bits}\overbrace{x_{3,2n} \cdots x_{3,1}x_{3,0}}^{2n+1\ bits}\underbrace{00\ldots00}_{n-1\ bits}$

And finally by applying property 2 we get:

$$\psi_4 = |-2^{n-1}x_3|_{2^{4n}-1}$$

$$= \underbrace{11\ldots11}_{n\ bits}\overbrace{\bar{x}_{3,2n} \cdots \bar{x}_{3,1}\bar{x}_{3,0}}^{2n+1\ bits}\underbrace{11\ldots11}_{n-1\ bits} \quad \textbf{(25)}$$

Correction Unit

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

$$Z = \tau + R \quad \textbf{(26)}$$

$$\tau = 2^n E$$

$$= \underbrace{E_{4n}E_{4n-1} \cdots E_1E_0\overbrace{00\ldots0}^{n\ bits}}_{5n+1\ bits} \quad \textbf{(27)}$$

$$R = \underbrace{R_nR_{n-1} \cdots R_1R_0}_{n+1\ bits} \quad \textbf{(28)}$$

Therefore, $\tau = \tau_{5n}\tau_{5n-1} \cdots \tau_1\tau_0$

$$Z = \underbrace{\tau_{5n}\tau_{5n-1} \cdots \tau_1\tau_0\overbrace{00\ldots00}^{4n\ bits}R_nR_{n-1} \cdots R_1R_0}_{5n+1\ bits} \quad \textbf{(29)}$$

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

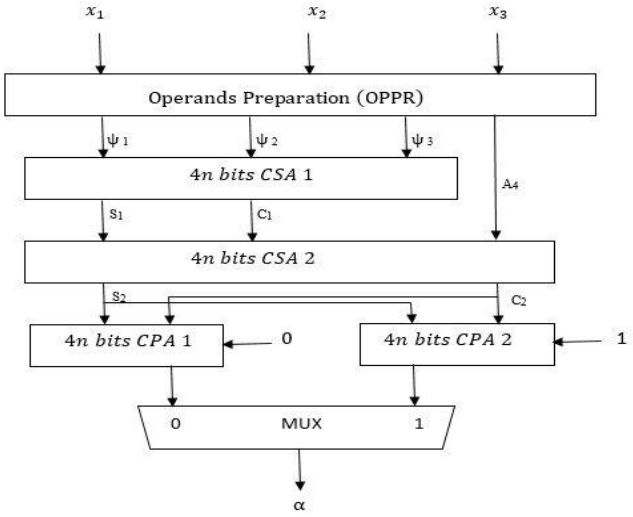

Figure 1: Block diagram of the partial reverse converter

A. Proposed Architecture

The output of the partial reverse converter α 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 CSAs 1 and 2 and two regular $4n$ bits CPAs 1 and 2. The results of these CPAs are passed on to a multiplexer (MUX 1) which would 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 corresponds to the decimal number X and α_y corresponds to the decimal number Y are added using a regular $(4n+1)$ bits CPA 3 in order to get E; at the same time, x_2 and y_2 are computed using a regular $(n+1)$ 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 $(n+1)$ A_{FA} of resources. Thus, the area requirement for the overflow detection component is $(5n+2)$ A_{FA}. Therefore, the total area requirement of the overflow detection scheme is $(37n+2)$ 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 $(5n+1)$ $bits$ CPA 5. The area requirement is $(5n+1)$ A_{FA} and its delay is also $(5n+1)$ D_{FA}.

The schematic diagrams for the proposed scheme are shown below:

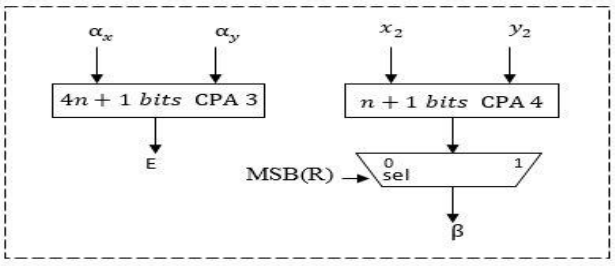

Figure 2: Overflow detection unit

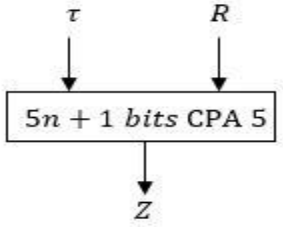

Figure 3: Correction unit

B. Numerical Illustrations

In this section, we present some numerical examples with the proposed scheme.

- From the moduli set $\{2^{2n}-1, 2^n, 2^{2n}+1\}$, when $n = 2$, we have {15, 4, 17}

Given two (2) numbers $X = 825$ and $Y = 500$, we then check for overflow in the sum of X and Y using moduli set$\{15,4,17\}$.

Thus

$$825 = (0,1,9)_{RNS(15|4|17)} = (0000, 0001, 01001)_{RNS(1111|100|10001)}$$

$$500 = (5,0,7)_{RNS(15|4|17)} = (0101, 0000, 00111)_{RNS(1111|100|10001)}$$

Therefore

$$825 + 500 = ((0000, 0001, 01001) + (0101, 0000, 00111))_{RNS(1111|100|10001)}$$

$$= (0101, 0001, 10000)_{RNS(1111|100|10001)}$$

RNS to decimal conversion of $(0101,0001,10000)_{RNS(1111|100|10001)}$ will result in the decimal number 305. Whilst the sum of the decimal numbers 825 and 500 is 1325. Therefore, a sign that overflow has occurred.

Checking for RNS overflow using the proposed algorithm

$\alpha_x = 206 = 011001110$

$\alpha_y = 125 = 001111101$

$E = 011001110 + 001111101 = 101001011$

$R = 001 + 000 = 001,\ \beta = 0$

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

- From the moduli set $\{2^{2n}\text{-}1, 2^n, 2^{2n}+1\}$, when $n = 3$, we have $\{63, 8, 65\}$

Given two (2) numbers $X = 7285$ and $Y = 16380$, we then check for overflow in the sum of X and Y using moduli set $\{63,8,65\}$. Thus

$$7285 = (40,5,5)_{RNS(63|8|65)} = (101000,0101,0101)_{RNS(111111|1000|1000001)}$$

$$16380 = (0,4,0)_{RNS(63|8|65)} = (0000,0100,0000)_{RNS(111111|1000|1000001)}$$

Therefore,

$$7285 + 16380 = ((101000,0101,0101) + (0000,0100,0000))_{RNS(111111|1000|1000001)}$$

$$= (101000,0001,0101)_{RNS(111111|1000|1000001)}$$

RNS to decimal conversion of $(101000,0001,0101)_{RNS(111111|1000|1000001)}$ will result in the decimal number 23665 which is the correct result of 7285 + 16380

Checking for RNS overflow using the proposed algorithm

$\alpha_x = 910 = 0001110001110$

$\alpha_y = 2047 = 0011111111111$

$E = 1110001110 + 11111111111 = 0101110001101$

$R = 0101 + 0100 = 1001,\ \beta = 1$

After processing, the scheme will obviously detect no overflow.

IV. PERFORMANCE EVALUATION

The architectural performance of the proposed scheme in terms of area and delay is presented and analyzed in this section. The result is compared with [4].Theoretical analysis from Table 1 shows that the proposed scheme has less delay compared to [4]. It is observed that as n increases, our scheme becomes faster thus requiring less delay. However, the higher hardware complexity of the proposed scheme can be explained by the larger dynamic range of the selected moduli set. From the AD^2 comparison it can be concluded that the proposed scheme will require less resources compared to the scheme proposed in [4].

Table 1. Area, Delay, AD^2 Comparison

SCHEME	[4]			PROPOSED		
n	Delay (22n+12)	Area (11n+6)	AD^2 ($5324n^3$ $+81712n^2+$ $4752n+864$)	Delay (12n+3)	Area (37n+2)	AD^2 ($5328n^3$ $+2952n^2+$ $477n+18$)
2	56	28	379808	27	76	55404
4	100	50	1668000	51	150	390150
6	144	72	4120992	75	224	1260000
8	188	94	7994336	99	298	2920698
10	232	116	13543584	123	372	5627988
12	276	138	21024288	147	446	9637614
14	320	160	30692000	171	520	15205320
16	364	182	42802272	195	594	22586850
18	408	204	57610656	219	668	32037948
20	452	226	75372704	243	742	43814358

V. CONCLUSION

The study has addressed the problem of overflow during addition operation in RNS which is seen to be one of the limitations that has confronted the full implementation of RNS in general purpose computing. The paper presented a generalized algorithm for detecting overflow and implemented the algorithm on the moduli set (2^{2n} - 1, 2^n, 2^{2n} + 1). The algorithms do not require full residue-to-binary conversion. The scheme is able to produce the correct result for the sum of two numbers whether overflow occurs or not. The moduli set used has a larger dynamic range which makes it more effective and efficient than well-known existing schemes.

REFERENCES

[1] 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

[2] 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.

[3] 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.

[4] H. Siewobr and K. A. Gbolagade (2014). RNS Overflow Detection by Operands Examination. International Journal of Computer Applications (0975 – 8887), Vol 85, No. 18 .

[5] 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.

[6] K.A. Gbolagade, R. Chaves, L. Sousa, and S.D. Cotofana (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.

[7] L. Theodore Houk (1989).Residue Addition Overflow Detection Processor. Boing Company, Seatle, Wash. Appl. No.:414276.

[8] 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.

[9] P. A. Agbedemnab 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 (975 – 8887)

[10] P. A. Mohan (2007). Reverse Converters for a New Moduli Set $\{2^{2n} - 1, 2^n, 2^{2n} + 1\}$. Circuit Systems Signal Processing, 26: 215.