Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 06, June-2023

Efficient Overflow Detection and Correction in RNS Addition using Partial Reverse Conversion: Exploring the Moduli Set $\{2^{2n-1}-1, 2^n, 2^n-1\}$

Mohammed I. Daabo
Department of Computer Science, School of Computing and Information Sciences,
C. K. Tedam University of Technology and Applied Sciences, Ghana

ABSTRACT-*This research paper presents a comprehensive algorithm for detecting and correcting overflow in Residue Number System (RNS) architecture. The underlying principle of the proposed algorithm relies on the Chinese Reminder Theorem, that enables the detection of overflow in RNS addition of two operands. Additionally, the scheme utilizes partial reverse conversion to identify overflow accurately and deliver error-free results. Subsequently, the algorithm is implemented using the moduli set $\{2^{2n-1}-1, 2^n, 2^n-1\}$. To demonstrate the effectiveness of this proposed scheme, a comparison profile is conducted and evaluated on the delay and area requirements. In all, the scheme showcases superior performance in terms of both AD^2(resources) and delay(speed), surpassing the capabilities of existing scheme. It is however observed that the new technique will impose high computational complexity (Area)*

Keywords: Residue Number System, reverse conversion, Chinese Reminder Theorem, Overflow, operands.

I. INTRODUCTION

Residue number system (rns) is an integer number system that utilizes remainders called residues to represent numbers. It supports parallelism, carry-free and borrow-free arithmetic and ensure single step multiplication without partial products. These unique characteristics make RNS particularly suitable for applications in the field of Digital Signal Processing (DSP), digital filtering, convolutions, correlations, Discrete Fourier Transforms (DFT), Fast Fourier Transforms (FFT) and Direct Digital Frequency Synthesis (DDFS) [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: $X = \left|\sum_{i=1}^{N} w_i x_i\right|_M$ **(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.

Overflow in computing refers to storing data that is larger than its designated memory location. Overflow is described in RNS as a condition in which a number falls outside the valid range of a specific RNS. A valid RNS number [4] is well represented by a number from the set [0. M-l]. Take, for example, the sum of the decimal numbers 90 and 25, which equals 115. The outcome of performing this addition in RNS with the moduli set {3, 5, 7} with dynamic range of 105 is $(1,0,3)_{RNS<3|5|7>}$. However, the number $(1,0,3)_{RNS<3|5|7>}$ is the decimal equivalent of 10. This is because the sum of 90 and 25, which is 115, is outside the legitimate range, therefore introducing 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 time, researchers have made considerable efforts to design efficient overflow detection schemes which are dependent on full reverse conversion [11]. Some proposed RNS overflow detection algorithms that 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. This research paper introduces a proposed scheme for detecting and correcting overflow in the Residue Number System (RNS) architecture. The proposed scheme employs partial reverse conversion, utilizing the Chinese Reminder Theorem (CRT) technique.

II. PROPOSED ALGORITHM

The proposed algorithm using CRT is presented in details in this section. The presented scheme 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)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 06, June-2023

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

Lemma 1 : For the given moduli set, we have

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

$$\left|M_2^{-1}\right|_{m_2} = \left|(2^{2n-1} - 1)(2^n - 1)^{-1}\right|_{2^n} \tag{4}$$

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

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

Lemma 2:

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

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

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

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

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

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

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

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

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

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

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

$$Z = 2^n(2^{4n} - 2 + 1) \tag{13}$$

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

$$Z = M$$

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:

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

(14)

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

$$\psi_2 = \left|-2^{3n}x_2\right|_{2^{4n}-1} \tag{16}$$

$$\psi_3 = \left|2^{3n-1}x_3\right|_{2^{4n}-1} \tag{17}$$

$$\psi_4 = \left|-2^{n-1}x_3\right|_{2^{4n}-1} \tag{18}$$

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:

$$x_{1,2n-1} \cdots x_{1,1}\, x_{1,0} \tag{19}$$

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

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

$$\psi_1 = \underbrace{r_{2,n}\cdots r_{2,1}r_{2,0}}_{n+1\text{ bits}}\ \overbrace{r_{2,(2n-1)}\cdots r_{2,1}r_{2,0}}^{2n\text{ bits}}\ \underbrace{r_{2,(2n-1)}\cdots r_{2,(n-1)}}_{n-1\text{ bits}} \tag{20}$$

which is a 4n bits residue number. **(20)**

Evaluation of ψ_2

The residue x_i can be represented as follows;

$$x_{2,n-1} \cdots x_{2,1}\, x_{2,0} \tag{21}$$

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

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:

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

And finally by applying property 2 we get:

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

$$= \bar{x}_{2,n-1} \cdots \bar{x}_{2,1}\bar{x}_{2,0}\overbrace{11\ldots11}^{3n\text{ bits}} \tag{22}$$

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

Evaluation of ψ_3 and ψ_4

The residue x_3 can be represented as follows;

$$x_{3,2n} \cdots x_{3,1}\, x_{3,0} \tag{23}$$

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

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

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

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 06, June-2023

By applying property 1:

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

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

And finally by applying property 2 we get:

$$\psi_4 = |-2^{n-1}x_3|_{2^{4n}-1} = \underbrace{11\ldots11}_{n\text{ bits}}\overbrace{\bar{x}_{3,2n}\cdots\bar{x}_{3,1}\bar{x}_{3,0}}^{2n+1\text{ bits}}\underbrace{11\ldots11}_{n-1\text{ bits}} \quad (25)$$

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

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

$$\tau = 2^n E = \underbrace{E_{4n}E_{4n-1}\cdots E_1E_0}_{5n+1\text{ bits}}\overbrace{00\ldots0}^{n\text{ bits}} \quad (27)$$

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

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

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

Implementation of equations **(26)** - **(29)** gives the correct result of Z whether overflow occurs or no

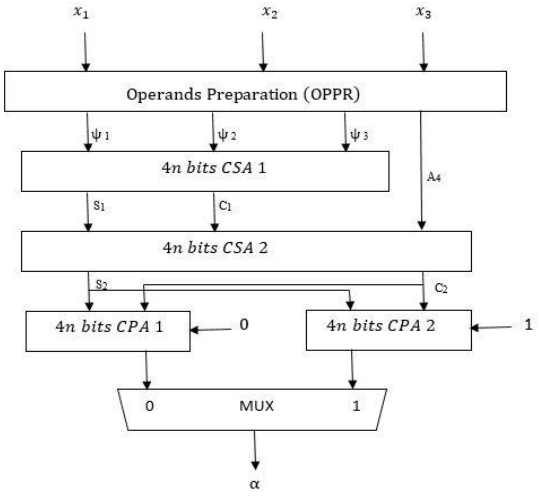

Figure 1: Block diagram of the partial reverse converter

IV. PROPOSED ARCHITECTURE

The output of the partial reverse converter α is determined using equation (14), where the parameters are defined in equations (15) to (18). The α values corresponding to numbers X and Y, denoted as α_x and α_y respectively, are computed using CSAs 1 and 2, along with regular 4n-bit CPAs 1 and 2. The results from these CPAs are passed to a multiplexer (MUX 1), which selects either CPA 1 or CPA 2 based on the carry out of CSA 1.

The α_x value corresponds to the decimal number X, and α_y corresponds to the decimal number Y. They are added using a regular (4n+1)-bit CPA 3 to obtain E. Simultaneously, x_2 and y_2 are computed using a regular (n+1)-bit CPA 4 to obtain R.

Another multiplexer (MUX 2) is used to set α as zero if the most significant bit (MSB) of R is 0, or as one (1) if the MSB of R is 1. This configuration is illustrated in figure 2, representing the overflow detection unit.

The CSAs 1 and 2 require an area of 4nAFA each, while CPAs 1 and 2 also require 4nAFA each. Therefore, the total area required to obtain α is 16nAFA. Consequently, for two numbers X and Y, the total area requirement is 32nAFA. CPA 3 requires an area of (4n+1)AFA, and CPA 4 requires (n+1)AFA. Hence, the area requirement for the overflow detection component is (5n+2)AFA. Therefore, the overall area requirement of the overflow detection scheme is (37n+2)AFA.

In terms of delay, each CSA (CSAs 1 and 2) introduces a delay of DFA, while each CPA (CPAs 1 and 2) imposes a delay of 4nDFA. Since they operate in parallel, for two numbers, the total delay becomes 8nDFA. Thus, the delay for computing α is (8n+2)DFA. The CPA pair 3 and 4 together impose a delay of (4n+1)DFA for the overflow detection unit. Therefore, the required delay for the proposed scheme is (12n+3)DFA.

The correction unit utilizes a regular (5n+1)-bit CPA 5. It requires an area of (5n+1)AFA and has a delay of (5n+1)DFA.

The schematic diagrams for the proposed scheme is provided below.

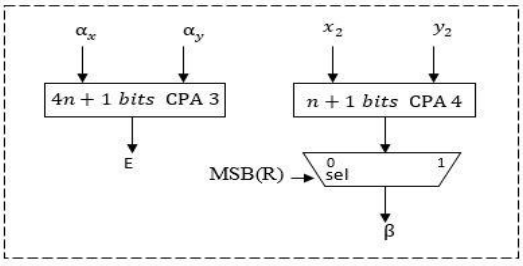

Figure 2: Overflow Detection

V. NUMERICAL ILLUSTRATIONS

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

From the moduli set $\{2^{2n-1}-1, 2^n, 2^n-1\}$, when $n = 2$, we have {7, 4, 3}.

Given two (2) numbers $X = 825$ and $Y = 500$, we then check for overflow in the sum of X and Y using moduli set {7, 4, 3}.

$\{2^{2n-1}-1, 2^n, 2^n-1\}$, where $m_1 = 2^{2n-1}-1$, $m_2 = 2^n$ and $m_3 = 2^n-1$ we have =

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

$$825 = (6,1,0)_{RNS(7|4|3)} = (110,001,000)_{RNS(111|100|011)}$$

$$512 = (1,0,2)_{RNS(7|4|3)} = (01,00,10)_{RNS(111|100|011)}$$

Therefore

$$825 + 512 = ((110,001,000) + (01,00,10))_{RNS(111|100|011)} = (111,001,010)_{RNS(1111|100|011)}$$

RNS to decimal conversion of $(111,001,010)_{RNS(111|100|011)}$ will result in the decimal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 06, June-2023

number 12432. Whilst the sum of the decimal numbers 825 and 500 is 1337. Clearly, this is a sign that overflow has occurred.

To check this RNS overflow, the following proposed algorithm is used.

Given that:

$$\alpha_x = 206 = 011001110$$
$$\alpha_y = 125 = 001111101$$
$$E = \alpha_x + \alpha_y$$
$$E = 011001110 + 001111101 = 101001011$$
$$R = 001 + 000 = 001,\ \beta = 0$$

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

From the moduli set $\{2^{2n-1}-1, 2^n, 2^n-1\}$, when n =3 , we have {31, 8, 7}

Given two (2) numbers $X = 7280$ and $Y = 16370$, we then check for overflow in the sum of X and Y using moduli set $\{31, 8, 7\}$. Thus

$$7280 = (26,0,0)_{RNS(31|8|7)} = (11010,0000,0000)_{RNS(11111|1000|111)}$$
$$16370 = (2,2,4)_{RNS(31|8|7)} = (10,10,100)_{RNS(11111|1000|111)}$$

Therefore,

$$7280 + 16370 = ((11010,0000,0000) + (10,10,100))_{RNS(11111|1000|111)}$$
$$= (110100,0010,0100)_{RNS(11111|1000|111)}$$

RNS to decimal conversion of $(101000,0001,0101)_{RNS(111111|1000|1000001)}$ will result in the decimal number 1225844 which is the correct result of 7285 + 16380.

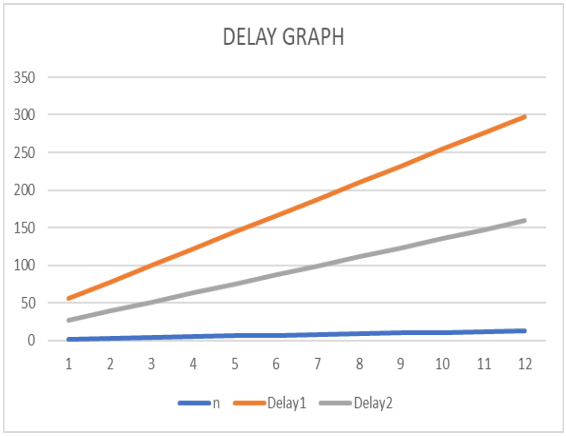

Figure 3: Delay graphs

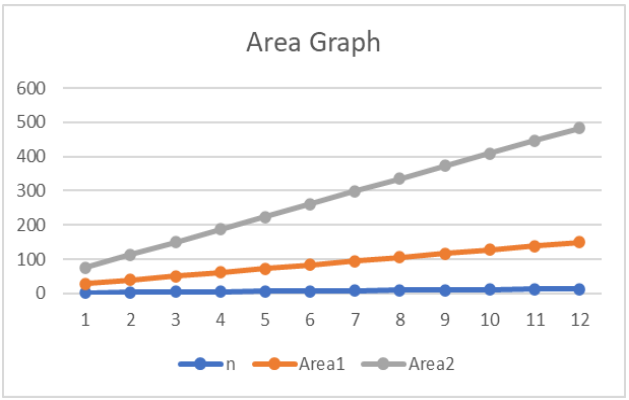

Figure 4: Area graphs

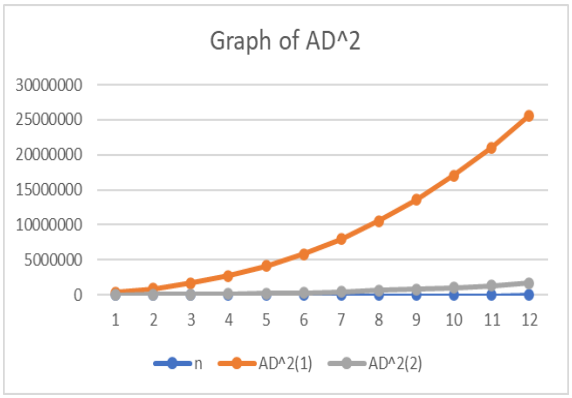

Figure 5: Graph of AD^2

VI. PERFORMANCE EVALUATION

This section presents and analyzes the architectural performance of the proposed scheme in terms of Area, Delay and AD^2. The results is compared with the proposal presented in [4]. From Table 1, the proposed method has less delay and AD^2 than [4] but presented a large architectural size (Area). It can be noted that as n increases, the proposed scheme becomes faster, necessitating less delay as shown in Figure 3. It is however observed in Figure 4 that there is a higher hardware complexity associated with the suggested technique and this can be attributed to the broader dynamic range of the moduli set chosen. The AD^2 comparison in Figure 5 shows that the suggested method will consume fewer resources than the scheme proposed in [4].

VII. CONCLUSION

This study focused on addressing the issue of overflow detection during addition operation in Residue Number System (RNS). This phenomenon has been one of the limiting factors that prevents RNS from being utilize in general-purpose computing. The research paper proposed a comprehensive algorithm for detecting overflow and applied it to the moduli set $\{2^{2n-1}-1, 2^n, 2^n-1\}$. Notably, these algorithm does not require a complete residue-to-binary conversion process. The proposed scheme ensures the accurate computation of the sum of two numbers, regardless of whether an overflow occurs or not. Moreover, the chosen moduli set offers a wider dynamic range, making the scheme more effective and efficient in terms of Delay and AD^2 but fall short in Area compared to the proposal in [4].

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 12 Issue 06, June-2023

VIII. 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.

[11] Salifu, A. (2021). New Reverse Conversion for Four-Moduli Set and Five-Moduli Set. *Journal of Computer and Communications*, 9, 57-66.

Table 1. Area, Delay and AD^2 Comparison

SCHEME	[4]			PROPOSED		
n	Delay1 (22n+12)	Area1 (11n+6)	$AD^2(1)$ ($5324n^3+81712n^2+4752n+864$)	Delay2 (12n+3)	Area2 (37n+2)	$AD^2(2)$ ($5328n^3+2952n^2+477n+18$)
2	56	28	379808	27	76	17036
3	78	39	894276	39	113	42381
4	100	50	1668000	51	150	83206
5	122	61	2732924	63	187	142703
6	144	72	4120992	75	224	224064
7	166	83	5864148	87	261	330481
8	188	94	7994336	99	298	465146
9	210	105	10543500	111	335	631251
10	232	116	13543584	123	372	831988
11	254	127	17026532	135	409	1070549
12	276	138	21024288	147	446	1350126
13	298	149	25568796	159	483	1673911