International Journal of Computer Applications (0975 – 8887)
Volume 110 – No. 16, January 2015

A Novel RNS Overflow Detection and Correction Algorithm for the Moduli Set $\{2^n-1, 2^n,\ 2^n+1\}$

P. A. Agbedemnab and E.K. Bankas
Department of Computer Science,
University for Development Studies,
Navrongo, Ghana.

ABSTRACT
In this paper, an efficient scheme for detecting and correcting overflow during addition in Residue Number System (RNS) is presented. The approach which is novel to the moduli set $\{2^n-1, 2^n,\ 2^n+1\}$ is based on the Chinese Remainder Theorem and demonstrates theoretically to be a very fast scheme compared to similar state of the art schemes. The proposed method is able to detect overflow in RNS addition without full reverse conversion; Additionally, the scheme also prevents the representation of wrong numbers as a result of overflow, thus the scheme gives the accurate result without errors whether overflow occurs or not. A comparison, which proves the efficiency of the proposed scheme, in terms of delay and area requirements is also presented.

General Terms
Residue Number System, Circuits and Systems, Computer Arithmetic, Computer Architecture, Overflow, Digital Signal Processing.

Keywords
Residue Number System, Chinese Remainder Theorem, overflow detection, overflow correction, moduli set

$\{2^n-1, 2^n,\ 2^n+1\}$.

1. INTRODUCTION
Residue Number System (RNS) is a non-weighted number system that utilizes remainders to represent numbers. This number system is capable of supporting parallel, carry-free and high speed arithmetic. The system is applied 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], [2], [3].

Nevertheless, operations such as division, overflow detection and correction, 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 [4].

RNS is determined by a set *S*, of *N* integers that are pair-wise relatively prime. That is

$$S = \{m_1, m_2, \dots, m_N\}$$

Where $gcd\,(m_i, m_j) = 1$ for $i, j = 1, \dots, N$ and $i \neq j$, and gcd means the greatest common divisor [8].

Every integer X in $[0, M-1]$ can be uniquely represented with a *N*-tuple where,

$$M = \prod_{i=1}^{N} m_i \quad, X \to (x_1, x_2, \dots, x_N)$$

and $\qquad x_i = |X|_{m_i} = (X \bmod m_i)$; 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.

Now, to calculate the number X from its residues, we can apply the CRT which is formulated as;

$$X = \left| \sum_{i=1}^{N} \ell_i \, |k_i x_i|_{m_i} \right|_M \qquad (\mathbf{1})$$

where,

$$M = \prod_{i=1}^{N} m_i \ ; \quad \ell_i = \frac{M}{m_i} \ ; \ |k_i \times \ell_i|_{m_i} = 1$$

Overflow is a condition where a number which falls outside the legitimate range of a particular RNS i.e.,$[0, M-1]$, $(M = \prod_{i=1}^{n} m_i)$ is well represented as a legitimate RNS number. During addition, this can be detected when the result of the addition is less than one of the addends. For example, given two RNS numbers X and Y such that $Z = X + Y\ mod\ M$ where $X \geq 0$ and $Y < M$, overflow will only occur when $Z < X$.

Another efficient way to detect overflow in RNS is via parity checking [1], [5]. 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. This technique is one of the best and fastest suggested methods to detect the overflow in RNS. However, this technique is only suitable for moduli sets with odd dynamic range (DR). But RNS systems that have even DR have more attractive features than those with odd DR. This is because using (2^n) modulo which tends dynamic ranges to even, greatly simplifies and reduces the delay and complexity of the scheme [4]. Thus the need to devise techniques of detecting overflow in moduli sets with even dynamic range.

Over the past few years, researchers have made considerable efforts to design overflow detection schemes to handle both odd and even dynamic ranges schemes which do not rely on the traditional techniques such as CRT and Mixed Radix Conversion (MRC) for full reverse conversion, recently proposed RNS overflow detection algorithms still rely on the later [4], and other costly and time consuming procedures such as base extension, group number and sign detection as in [5], [6] and [7]. The scheme in [6] is demonstrated to be better than those in [4] and [5] in terms of both area and delay.

In this paper, an efficient algorithm for RNS overflow detection and correction for the moduli set $\{2^n-1, 2^n,\ 2^n+1\}$ is proposed. The proposal does not require full reverse conversion and is suitable for even dynamic range schemes. It is a very fast scheme compared to best known similar state of the art designs.

The rest of the paper is organized as follows: Section 2 presents the proposed method. In Section 3, the hardware implementation of the proposed scheme is presented, a simplified algorithm with numerical examples are also presented. The performance of the proposed scheme is evaluated in Section 4 whiles the paper is concluded in Section 5.

2. PROPOSED METHOD

In this section, a new method for detecting overflow as well as preventing the representation of illegitimate numbers as if they are legitimate numbers in the DR (thus correcting overflow) is presented.

2.1 Algorithm for the Proposed Scheme

The algorithm for the proposed method is as follows;

1. Compute ρ_x and ρ_y according to (7)
2. Determine K and β according to (10) and (11)
3. *Overflow occurs only under one of the following conditions;*
   (i) If the MSB of K i.e $K_{2n} = 1$
   (ii) If K_{2n-1} down to K_0 is "1"
   (iii) If K_{2n-1} down to K_1 is "1" and $\beta = 1$
4. 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 $\{2^n - 1, 2^n, 2^n + 1\}$, where $m_1 = 2^n - 1$, $m_2 = 2^n$ and $m_3 = 2^n + 1$ we have

$$\ell_1 = 2^n(2^n + 1);\ \ell_2 = (2^{2n} - 1);$$

$$\ell_3 = 2^n(2^n - 1) \qquad \textbf{(2)}$$

Theorem 1*:* For the given moduli set, we have

$$|k_1|_{m_1} = |2^{n-1}|_{m_1} \qquad \textbf{(3)}$$

$$|k_2|_{m_2} = |-1|_{m_2} \qquad \textbf{(4)}$$

$$|k_3|_{m_3} = |-2^{n-1}|_{m_3} \qquad \textbf{(5)}$$

The proof of (3) – (5) is demonstrated in [9].

Theorem 2: For the given moduli set, any RNS number X can be represented as;

$$X = 2^n\rho + x_2 \qquad \textbf{(6)}$$

where,

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

Proof: Substituting equations (2) through to (5) into (1) and factorizing out 2^n we obtain (6).

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

$$X = 2^n\rho_x + x_2 \qquad \textbf{(8)}$$

$$Y = 2^n\rho_y + y_2 \qquad \textbf{(9)}$$

$$Z = 2^n(\rho_x + \rho_y) + x_2 + y_2$$

$$= 2^n K + \mu \qquad \textbf{(10)}$$

Where $K = \rho_x + \rho_y$ and $\mu = x_2 + y_2$

Let

$$\beta = \begin{cases} \mu < 2^n, & 0 \\ \mu \geq 2^n, & 1 \end{cases} \qquad \textbf{(11)}$$

Theorem 3: Given any two RNS numbers

$X = (x_1, x_2, x_3)$ and $Y = (y_1, y_2, y_3)$, overflow occurs if and only if

$$K \geq 2^{2n} - 1 \qquad \textbf{(12)}$$

or

$$K = 2^{2n} - 2 \text{ and } \beta = 1 \qquad \textbf{(13)}$$

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

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

$$\geq M + \mu$$

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

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

$$= M,$$

which is also outside the legitimate range, therefore overflow will occur. Hence proofed.

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 K should be added to the DR to be $[0, M + K - 1]$ in order to legitimize Z.

3. HARDWARE IMPLEMENTATION

Equation (7) can further be simplified as follows

$$\rho = |\varphi_1 + \varphi_2 + \varphi_3 + \varphi_4|_{2^{2n}-1} \qquad \textbf{(14)}$$

where

$$\varphi_1 = |(2^n x_1 + x_1)2^{n-1}|_{2^{2n}-1} \qquad \textbf{(15)}$$

$$\varphi_2 = |-2^n x_2|_{2^{2n}-1} \qquad \textbf{(16)}$$

$$\varphi_3 = |-x_3|_{2^{2n}-1} \qquad \textbf{(17)}$$

$$\varphi_4 = |2^{n-1}x_3|_{2^{2n}-1} \qquad \textbf{(18)}$$

Now, we consider (14)-(17) and simplify them for implementation in a VLSI system. It is necessary to note that $x_{i,j}$ means the *j*-th bit of x_i.

Evaluation of φ_1
The residue x_1 can be represented as follows;

$$x_1 = x_{1,n-1} \ldots x_{1,1}x_{1,0} \qquad \textbf{(19)}$$

Thus,

$$|(2^n x_1 + x_1)2^{n-1}|_{2^{2n}-1} =$$

$$\left| 2^{n-1} \left(\underbrace{x_{1,n-1} \ldots x_{1,0} \overbrace{0 \ldots 0}^{n\ Bits}}_{2n-bits} + \underbrace{\overbrace{0 \ldots 0}^{n\ Bits} x_{1,n-1} \ldots x_{1,0}}_{2n} \right) \right|_{2^{2n}-1}$$

$$= \left|2^{n-1}\left(\underbrace{x_{1,n-1}\ldots x_{1,1}x_{1,0}x_{1,n-1}\ldots x_{1,1}x_{1,0}}_{2n-bits}\right)\right|_{2^{2n}-1}$$

$$= \overbrace{x_{1,0}x_{1,n-1}\ldots x_{1,1}x_{1,0}}^{n+1\ Bits}\underbrace{x_{1,n-1}\ldots x_{1,n+2}x_{1,1}}_{n-1} \qquad \textbf{(20)}$$

Evaluation of φ_2*:*
The residue x_2 can be represented as follows;

$$x_2 = x_{2,n-1}\ldots x_{2,1}x_{2,0} \qquad \textbf{(21)}$$

Therefore,

$$|-2^n x_2|_{2^{2n}-1} = \bar{x}_{2,n-1}\ldots\bar{x}_{2,1}\bar{x}_{2,0}\overbrace{11\ldots11}^{n\ Bits} \qquad \textbf{(22)}$$

Evaluation of φ_3 *and* φ_4:
The residue x_3 can be represented as follows;

$$x_3 = x_{3,n}\ldots x_{3,1}x_{3,0} \qquad \textbf{(23)}$$

Therefore,

$$\varphi_3 = |-x_3|_{2^{2n}-1} = \overbrace{11\ldots11}^{n\ Bits}\underbrace{\bar{x}_{3,n-1}\ldots\bar{x}_{3,1}\bar{x}_{3,0}}_{n\ Bits} \qquad \textbf{(24)}$$

Again,

$$\varphi_4 = |2^{n-1}x_3|_{2^{2n}-1} = \underbrace{0x_{3,n}\ldots x_{3,1}x_{3,0}}_{n+1\ Bits}\overbrace{00\ldots00}^{n-1\ Bits} \qquad \textbf{(25)}$$

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

$$Z = \tau + \mu \qquad \textbf{(26)}$$

$$\tau = 2^n K = \underbrace{K_{2n}K_{2n-1}\ldots K_1K_0\overbrace{00\ldots0}^{n\ bits}}_{3n+1\ bits} \qquad \textbf{(27)}$$

$$\mu = \underbrace{\mu_n\mu_{n-1}\ldots\mu_1\mu_0}_{n+1\ bis} \qquad \textbf{(28)}$$

Therefore,

$$Z = \underbrace{\tau_{3n}\tau_{3n-1}\ldots\tau_1\tau_0 + \overbrace{00\ldots0}^{2n\ bits}\mu_n\mu_{n-1}\ldots\mu_1\mu_0}_{3n+1\ bits} \qquad \textbf{(29)}$$

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

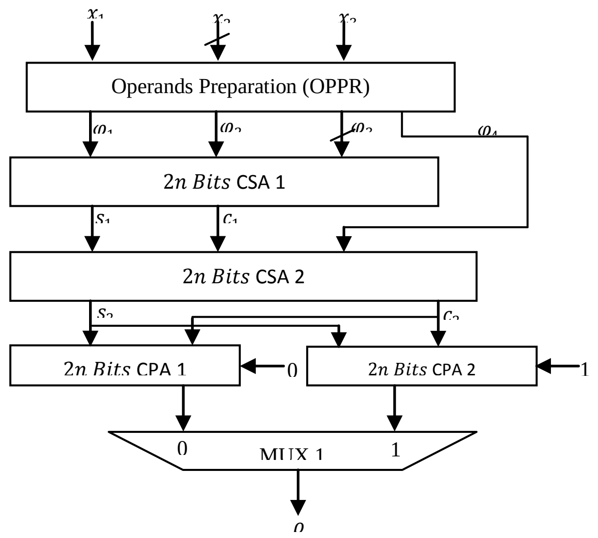

Fig 1: Block diagram of the partial reverse converter

3.1 Proposed Architecture

ρ is computed according to equation (14) where all the parameters are defined in equations (15) – (18). For two numbers X and Y, ρ_x and ρ_y are the ρ values corresponding X and Y respectively and are computed separately. As shown in *Fig* 1, ρ is computed using CSAs 1and 2 and two regular $2n$-bit 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.

ρ_x corresponding to the binary number X and ρ_y corresponding to the binary number Y is added using a regular $(2n+1)$ $bits$ CPA 3 in order to get K; at the same time, x_2 and y_2 is computed using a regular $(n+1)$ $bits$ CPA 4 to obtain μ. A multiplexer (Mux 2) is used to select the value of β to be zero if the most significant bit (MSB) of μ is 0, otherwise, it selects one (1) if the MSB of μ is 1. This is shown in figure 2 which is the overflow detection unit.

CSAs 1 and 2 require an area of $2n\Delta_{FA}$ each as well as CPAs 1 and 2. Therefore, in order to obtain ρ will require a total area of $8n\Delta_{FA}$. So for two numbers X and Y, the total area requirement will be $16n\Delta_{FA}$.

CPA 3 demands an area of $(2n+1)\Delta_{FA}$ and CPA 4 also requires $(n+1)\Delta_{FA}$ of resources. Thus, the area requirement for the overflow detection component is $(3n+2)\Delta_{FA}$. Therefore, the total area requirement of the overflow detection scheme is $(19n+2)\Delta_{FA}$.

Regarding the delay, each CSA (i.e. CSAs 1 and 2) impose a delay of D_{FA} while the CPA pair 1 and 2 impose a delay of $2nD_{FA}$ since they are in parallel, for two numbers this will become $4nD_{FA}$, thus delay imposed on computing ρ is $(4n+2)D_{FA}$. Also the CPA pair 3 and 4 impose a delay of $(2n+1)D_{FA}$for the overflow detection unit. Therefore, the delay required for the proposed scheme is $(6n+3)D_{FA}$.

The correction unit uses a regular $(3n+1)$ $bits$ CPA 5. The area requirement is $(3n+1)\Delta_{FA}$ and its delay is also $(3n+$ *1)DFA.*

The schematic diagrams for the proposed scheme are shown below.

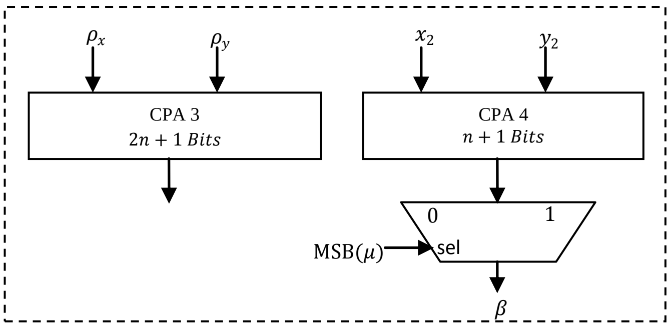

Fig 2: Overflow detection unit

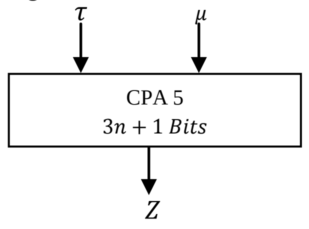

Fig 3: Correction unit

3.2 Numerical Illustrations

Let us now look at numerical examples with the proposed scheme.

Checking overflow in the sum of 49 and 21 using RNS moduli set {3, 4, 5}

$$49 = (1,1,4)_{RNS\langle 3|4|5\rangle} = (01,01,100)_{RNS\langle 11|100|101\rangle}$$

$$21 = (0,1,1)_{RNS\langle 3|4|5\rangle} = (00,01,001)_{RNS\langle 11|100|101\rangle}$$

$$= \big((01,01,100) + (00,01,001)\big)_{RNS\langle 11|100|101\rangle}$$

$$= (01,10,000)_{RNS\langle 11|100|101\rangle}$$

RNS to decimal conversion of $(01,10,000)_{RNS\langle 11|100|101\rangle}$ will result in the decimal number 10. Whilst the sum of the decimal numbers 49 and 21 is 70 which is obvious of overflow occurring.

Checking for RNS overflow using the proposed algorithm

$$\rho_x = 12 = 01100$$

$$\rho_y = 5 = 00101$$

$$K = \rho_x + \rho_y = 01100 + 00101 = 10001$$

$$\mu = 01 + 01 = 010, \ \ \beta = 0$$

Since the MSB of K is “1”, the scheme will detect that overflow has occurred.

From (27),

$$Z = 1000100 + 0000010 = 1000110 = (70)_{\text{decimal}}$$

Checking overflow in the sum of 28 and 32 using RNS moduli set {3, 4, 5}

$$28 = (1,0,3)_{RNS\langle 3|4|5\rangle} = (01,00,011)_{RNS\langle 11|100|101\rangle}$$

$$32 = (2,0,2)_{RNS\langle 3|4|5\rangle} = (10,00,010)_{RNS\langle 11|100|101\rangle}$$

$$= \big((01,00,011) + (10,00,010)\big)_{RNS\langle 11|100|101\rangle}$$

$$= (00,00,000)_{RNS\langle 11|100|101\rangle}$$

RNS to decimal conversion of $(00,00,000)_{RNS\langle 11|100|101\rangle}$ will result in the decimal number 0. Whilst the sum of the decimal numbers 28 and 32 is 60, a clear sign of overflow occurring.

Checking for RNS overflow using the proposed algorithm

$$\rho_x = 7 = 00111$$

$$\rho_y = 8 = 01000$$

$$K = \rho_x + \rho_y = 00111 + 01000 = 01111$$

$$\mu = 00 + 00 = 000, \ \ \beta = 0.$$

Even though, the MSB of K is not “1”, all other bits are “1” therefore the scheme will detect that overflow has occurred.

From (27),

$$Z = 0111100 + 0000000 = 0111100 = (60)_{\text{decimal}}$$

Checking overflow in the sum of 10 and 11 using RNS moduli set {3, 4, 5}

$$10 = (1,2,0)_{RNS\langle 3|4|5\rangle} = (01,10,000)_{RNS\langle 11|100|101\rangle}$$

$$11 = (2,3,1)_{RNS\langle 3|4|5\rangle} = (10,11,001)_{RNS\langle 11|100|101\rangle}$$

$$= \big((01,10,000) + (10,11,001)\big)_{RNS\langle 11|100|101\rangle}$$

$$= (00,01,001)_{RNS\langle 11|100|101\rangle}$$

RNS to decimal conversion of $(00,01,001)_{RNS\langle 11|100|101\rangle}$ will result in the decimal number 21 which is correct result of 10 + 11.

Checking for RNS overflow using the proposed algorithm

$$\rho_x = 2 = 00010$$

$$\rho_y = 2 = 00010$$

$$K = 00010 + 00010 = 00100$$

$$\mu = 10 + 11 = 101, \ \ \beta = 1.$$

After processing, the scheme will obviously detect no overflow since it is only $K_{2n-2} = 1$.

And from (27),

$$Z = 0010000 + 0000101 = 0010101 = (21)_{\text{decimal}}$$

4. PERFORMANCE EVALUATION

In order to evaluate the performance of the proposed overflow detection scheme, it is compared with similar best known state of the art schemes.

Theoretical analysis from Table 1 shows that the proposed scheme has less delay and complexity without compromising on accuracy compared to [4] which is the current best state of the art and has a correction component. The proposed scheme is also faster than [6] which was the best state of the art before [4]. Even though, the hardware complexity of the proposed scheme is higher than that in [6], the proposed scheme uses three comparators and a single AND gate whilst [6] uses six comparators and three AND gates which are not included in the comparison. It is also clear from the AD^2 analysis that the proposed scheme is very efficient than the state of the art schemes.

The correction part is not included in the evaluation just as it is not included in [4] for fairness. In any case, the accurate result is a $(3n+1)\ bit$ sum (Z) in (10) therefore, a $(3n+1)\ bit$ adder is designed to cater for the addition.

5. CONCLUSION

Detecting overflow is one of the most important and complex operations in RNS. In this paper, a novel method for detecting and correcting overflow during addition is presented. The proposed technique does not require full RNS-binary conversion. The proposed scheme is able to give the correct result for the sum of two numbers whether overflow occurs or not. The proposed scheme is demonstrated theoretically to be very efficient than similar state of the art designs. Since only theoretical analysis was presented, our next focus will be to implement the proposed method using VHDL in Xilinx.

6. REFERENCES

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

[2] K. A. Gbolagade. 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, April 2013.

[3] K.A. Gbolagade, R. Chaves, L. Sousa, and S.D. Cotofana. 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, Paris, France, June,2010.

[4] D. Younes and P. Steffan. 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, 2013.

[5] M. Rouhifar, M. Hosseinzadeh, S. Bahanfar and M. Teshnehlab. Fast Overflow Detection in Moduli set $\{2^{n}-1, 2^{n}, 2^{n}+1\}$. International Journal of Computer Science Issues, Vol (8/3), pp 407-414, May 2011.

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

[7] M. Hosseinzadeh, A.S. Molahosseini and K. Navi. A parallel Implementation of the Reverse Converter for the moduli set $\{2^{n}-1, 2^{n}, 2^{n-1}-1\}$. World Academy of Science, Engineering and Technology, Vol. 55, pp. 494-498. 2009.

[8] A. S. Molahosseini, K. Navi. New Arithmetic residue to binary Converters. International Journal of Computer Sciences and Engineering Systems, Vol. 1, No.4, pp. 295-299 Oct., 2007.

[9] E. K. Bankas and K. A. Gbolagade. A New Efficient FPGA Design of Residue-To-Binary Converter. International Journal of VLSI design & Communication Systems (VLSICS), Vol 4, No. 6, December, 2013.

Table 1: Area, Delay Comparison

Scheme	**Area**(Δ_{FA})	**Delay**(D_{FA})	$\boldsymbol{AD^2}$
[6]	$11n+6$	$22n+12$	$5324n^3+81712n^2+4752n+864$
[4]	$37n+18$	$16n+\log n+13$	$9472n^3+20000n^2+13661n+3042$
Proposed	$19n+2$	$6n+3$	$684n^3+756n^2+243n+18$