International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 08 | Aug 2018 www.irjet.net p-ISSN: 2395-0072

A HYBRID RESIDUE TO BINARY CONVERTER FOR THE MODULI SET

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

Mohammed I. Daabo (PhD)

Department of Computer Science, University for Development Studies, Navrongo, Ghana.

Abstract - *Residue Number System (RNS) is a non-weighted number systems with attractive features for modern day digital systems and computations. This number system has however not found universal usage in digital computing due to some challenges such as converting from decimal/binary numbers to their RNS equivalent representation and vice visa. A lot of work has been done on reverse conversion mostly using the traditional methods based on the Chinese Remainder Theorem (CRT) and the Mixed Radix Conversion (MRC). This paper presents a fast residue to binary converter based on the cyclic jump technique. The proposed conversion algorithm is based on the cyclic pattern inherent in residue number system. By the cyclic pattern property, each residue sequence of modulus m_i has a period of m_i entries defined on the residue table. The algorithm is defined to make a maximum of N consecutive jumps (j_i) in the residue table such that each location has one more zero residue than the previous location and that the final location must be (0, 0, 0). The algorithm is proposed on a generalized 3-moduli set and implemented on* $\{2^{n+1}-1, 2^n+1, 2^n-1\}$. *It is observed that the method is mainly based on modular computations and generates smaller numbers. It is further observed that parameters such as multiplicative inverse and big M which often increase the computational time of most existing techniques are absent in the proposed scheme. Theoretical analysis shows that the proposed scheme is more efficient and outperformed similar existing schemes in terms of area and delay.*

Key Words: **Residue Number System (RNS), Binary Converter, Modular computations, Cyclic pattern, CRT, MRC**

1. INTRODUCTION

Residue Number System (RNS) is inherently a carry-free number system useful for systems that involve large computations such as Digital Signal Processing (DSP) and Fourier Transforms, [1], [2]. There is a great interest in RNS because a great deal of computing now takes place in embedded processors, such as those found in mobile devices which normally require high speed and low-power consumption. The absence of carry-propagation facilitates the realization of high-speed, low-power arithmetic. Also, computer chips are now becoming so dense that full testing will no longer be possible and therefore, makes fault-tolerance and the general area of computational integrity essential, [3]. Even though, there have been some progress in the implementation of some difficult arithmetic operations such as division, number/data conversion, scaling, overflow and magnitude detections over the years, much still needs to be done in order to achieve general purpose usage and implementation of RNS processors [4], [5]. An RNS number$X$, is represented as $x_i = |X|_{m_i}$, where $m_i = \{m_1, m_2 \dots m_n\}$, a set of pairwise relatively prime integers such that $m_1 \neq m_2 \neq \cdots \neq m_n$ and $\gcd(m_1, m_2), \dots, \gcd(m_{n-1}, m_1) = 1$. The residue set $x_i = [x_1, x_2, \dots, x_n]$ is uniquely represented provided X lies within the legitimate range $[0, M-1]$ where $M = \prod_{i=1}^{n} m_i$ is the Dynamic, [5]. The conversion of an RNS number into its decimal/binary equivalent number (*a process called reverse conversion*) has long been mainly based on the Chinese Remainder Theorem (CRT) and the Mixed Radix Conversion (MRC) techniques with few modifications being their variants of recent times. Whiles the former deals with the modulo-M operation, the later does not but computes sequentially which tends to reduce the complexity of the architecture, [6]. Another method presented in [7] used a proposed conversion algorithm based on the cyclic pattern inherent in residue number system to perform reverse conversion. Recently, some techniques have been developed with semblance of reverse conversion process; [8] proposed an algorithm to detect overflow in the moduli set $(2^n-3, 2^n-1, 2^n, 2^n+1, 2^n+3)$ using a parity checking technique but used ROMs for implementation. They adopted a reverse converter which used the CRT technique. In [2], a partial converter was proposed and used for the moduli set $(2^{2n+1}-1, 2^n+1, 2^n-1)$. This converter was then used to detected overflow. All these schemes either relied on complete reverse conversion process as in the case of [10] or other costly and time consuming procedures such as base extension, group number and sign detection as in [9] and [11].

In this paper, a hybrid method of the Cyclic Jump Method which encapsulates the Mixed Radix Conversion Method for the moduli set $\{2^{n+1}-1, 2^n+1, 2^n-1\}$ is presented. The method applies to moduli sets with common factors and those without common factors as well. The rest of the paper is organised as follows: Section 2 presents the proposed method with its hardware implementation in Section 3. Section 4 is the hardware realization of the proposed scheme with a

schematic diagram with numerical illustrations in Section 5. The performance of the scheme is presented in Section 6 whiles Section 7 concludes the paper.

2. PROPOSED METHOD

Given the moduli set$\{2^{n+1}-1, 2^n+1, 2^n-1\}$, where, $m_1 = 2^{2n+1}-1$, $m_2 = 2^n+1$ and $m_3 = 2^n-1$, a hybrid Cyclic Jump Method which encapsulates the Mixed Radix Conversion Method is employed in the following algorithm to achieve the conversion process.

Given the RNS number $X = (r_1, r_2, r_3)$

(i) A first jump, J_1 is defined which corresponds to the first residue in X. i.e. $J_1 = r_1$.
This is followed by the determination of a first location, L_1 which reduces the first residue to zero and is defined as $L_1 = X - J_1$. Thus,

$$L_1 = r_1 - J_1 = \begin{bmatrix} |r_1 - J_1|_{2^{n+1}-1} = r_1' \\ |r_2 - J_1|_{2^n} = r_2' \\ |r_3 - J_1|_{2^n-1} = r_3' \end{bmatrix} \quad (1)$$

(ii) The second jump, J_2 is defined such that
$J_2 = (2^{n+1}-1)K_2$ and
$|r_2' - J_2|_{2^n} = 0 \Rightarrow |r_2' - (2^{n+1}-1)K_2|_{2^n} = 0$ (2)

Similarly, the second location L_2 is determined by:

$$L_2 = L_1 - J_2 = \begin{bmatrix} |r_1' - J_2|_{2^{n+1}-1} = r_1'' \\ |r_2' - J_2|_{2^n} = r_2'' \\ |r_3' - J_2|_{2^n-1} = r_3'' \end{bmatrix} \quad (3)$$

(iii) Also, since the moduli set comprises of three moduli, a third jump, J_3 is defined such that
$J_3 = (2^{n+1}-1)(2^n)K_3$ and
$|r_3'' - J_3|_{2^n-1} = 0$
$\Rightarrow |r_3'' - (2^{n+1}-1)(2^n)K_3|_{2^n-1} = 0$ (4)

And, the third location, L_3 then determined by:

$$L_3 = L_2 - J_3 = \begin{bmatrix} |r_1'' - J_3|_{2^{n+1}-1} = r_1''' \\ |r_2'' - J_3|_{2^n} = r_2''' \\ |r_3'' - J_3|_{2^n-1} = r_3''' \end{bmatrix} \quad (5)$$

At this point, $(r_1''', r_2''', r_3''') = (0, 0, 0)$ which

signifies an end to the jumps.

(iv) Finally, the decimal/binary number X is the result of summing the J_i's, thus,

$$X = J_1 + J_2 + J_3 \quad (6)$$

3. HARDWARE IMPLEMENTATION

For the given moduli set, where $m_1 = 2^{n+1}-1, m_2 = 2^n, m_1 = 2^n - 1$;

$$J_1 = r_1$$

$$= r_{1,n} r_{1,n-1} \dots r_{1,1} r_{1,0} \quad (7)$$

$$L_1 = \begin{bmatrix} |r_1 - r_1|_{m_1} \\ |r_2 - r_1|_{m_2} \\ |r_3 - r_1|_{m_3} \end{bmatrix} = \begin{bmatrix} \overbrace{00\dots00}^{n+1-bits} \\ \left|r_{2,n-1}\dots r_{2,1}r_{2,0} + \overline{r_{1,n}\dots r_{1,1}r_{1,0}}\right|_{2^n} \\ \left|r_{3,n-1}\dots r_{3,1}r_{3,0} + \overline{r_{1,n}\dots r_{1,1}r_{1,0}}\right|_{2^n-1} \end{bmatrix}$$

$$= \begin{bmatrix} \overbrace{00\dots00}^{n+1-bits} \\ r'_{2,n-1}\dots r'_{2,1}r'_{2,0} \\ r'_{3,n-1}\dots r'_{3,1}r'_{3,0} \end{bmatrix} \quad (8)$$

$$J_2 = m_1K_2$$

$$= 2^{n+1}K_2 - K_2$$

$$= K_{2,n-1}\dots K_{2,0}\ \overbrace{00\dots0}^{n+1-bits} + \overbrace{11\dots1}^{n+1-bits}\ \bar{K}_{2,n-1}\dots\bar{K}_{2,0}$$

$$= J_{2,2n}J_{2,2n-1}\dots J_{2,1}J_{2,0} \quad (9)$$

where,

$$K_2 = \left||r_2 - r_1|m_1^{-1}\right|_{m_2}$$

$$= \left||r_2 - r_1|(-1)\right|_{2^n}$$

$$= |r_1 - r_2|_{2^n}$$

$$= K_{2,n-1}\dots K_{2,1}K_{2,0} \quad (10)$$

$$L_2 = \begin{bmatrix} 0 \\ |r'_2 - J_2|_{m_2} \\ |r'_3 - J_2|_{m_3} \end{bmatrix} = \begin{bmatrix} \overbrace{00\dots00}^{n+1-bits} \\ \overbrace{00\dots00}^{n-bits} \\ \left|r'_{3,n-1}\dots r'_{3,1}r'_{3,0} + \bar{J}_{2,2n}\dots\bar{J}_{2,1}\bar{J}_{2,0}\right|_{2^n-1} \end{bmatrix}$$

$$= \begin{bmatrix} \overbrace{00\dots00}^{n+1-bits} \\ \overbrace{00\dots00}^{n-bits} \\ r''_{3,n-1}\dots r''_{3,1}r''_{3,0} \end{bmatrix} \quad (11)$$

$$J_3 = m_1m_2K_3$$

$$= 2^{2n+1}K_3 - 2^nK_2$$

$$= K_{3,n-1}\dots K_{3,0}\ \overbrace{0\dots00}^{(2n+1)-bits} - \left(\overbrace{00\dots0}^{(n+1)-bits}\ K_{3,n-1}\dots K_{3,0}\ \overbrace{0\dots0}^{n-bits}\right)$$

$$= K_{3,n-1}\dots K_{3,0}\ \overbrace{0\dots00}^{(2n+1)-bits} + \left(\overbrace{1\dots1}^{(n+1)-bits}\ K_{3,n-1}\dots K_{3,0}\ \overbrace{1\dots1}^{n-bits}\right)$$

$$= J_{3,3n}J_{3,3n-1}\dots J_{3,1}J_{3,0} \quad (12)$$

where,

$$K_3 = \left|(r_3 - r_1)|m_1^{-1}|_{m_3} - K_2)|m_2^{-1}|_{m_3}\right|_{m_3}$$

$$= |r_3 - r_1 - K_2|_{2^n-1}$$

$$= \left|r_{3,n-1} \dots r_{3,1} r_{3,0} + \bar{r}_{1,n} \dots \bar{r}_{1,1} \bar{r}_{1,0} + \bar{K}_{2,n-1} \dots \bar{K}_{2,1} \bar{K}_{2,0}\right|_{2^n-1}$$

$$= K_{3,n-1} K_{3,n-2} \dots K_{3,1} K_{3,0} \quad (13)$$

Finally,

$$X = J_1 + J_2 + J_3$$

$$= \underbrace{\overbrace{0 \dots 0}^{2n-bits}\ r_{1,n} \dots r_{1,0} + \overbrace{0 \dots 0}^{n-bits} J_{2,2n} \dots J_{2,0} + J_{3,3n} \dots J_{3,0}}_{(3n+1)-bits} \quad (14)$$

3. HARDWARE REALISATION

The hardware architecture of the proposed scheme is first realised by computing the first location, L_1 according to equation (7) since J_1, the first jump is the same as the residue corresponding to the first moduli. This is achieved by an array of three complementary adders: CADD1, CADD2 and CADD3 computing parallelly. Whilst the result from CADD1 results in an $(n+1)$-bit zero number, the result from CADD2 is complemented to get K_2 from which J_2 (the second jump) is gotten according to equation (9) with the aid of another complementary adder, CADD4. J_2 is necessary for the computation of L_2; CADD5 and CADD6 are used to achieve that according to equation (11). In order to get K_3, a carry propagate adder (CPA1) is used to add the save and carry from the carry save adder (CSA1) according to equation (13). Another complementary adder CADD7 is then used to get J_3 according to equation (12). Finally, values of the three jumps are added using CSA2 where the carry c_2 and save, s_2 are propagated using CPA2.

The complementary and propagate adders used are modulo adders and so do not impose repetitive delays on the scheme. The results from CADD1 and CADD5 are known to be zeros and thus can be hardwired instead of employing physical adders; in such case, no hardware complexities will be imposed therein. Also, the delay imposed by the carry save adders is unity. However, the hardware complexities of the scheme are measured according to the modulo units used.

Thus, the hardware complexities and delay (time required for processing) of the proposed scheme are estimated as follows;

$$Area = A_{CADD2} + A_{CADD3} + A_{CSA1} + A_{CADD4}$$

$$+A_{CPA1} + A_{CADD6} + A_{CADD7} + A_{CSA2} + A_{CPA2}$$

$$= 7n\Delta_{FA} + 2(3n+1)\Delta_{FA}$$

$$= (13n+2)\Delta_{FA}$$

$$Delay = D_{CADD2} + D_{CADD4} + D_{CPA1} + D_{CADD7}$$

$$+D_{CSA2} + D_{CPA2}$$

$$= 4nD_{FA} + D_{FA} + (3n+1)D_{FA}$$

$$= (7n+2)D_{FA}$$

The schematic diagram of the proposed scheme is shown in Fig 1.

4. NUMERICAL ILLUSTRATIONS

For the given the moduli set $\{2^{n+1}-1, 2^n, 2^n-1\}$ and taking $n = 2$, let the residue set for the decimal number X, be $(1,3,2)$. Thus, $m_1 = 7$, $m_2 = 4$, $m_3 = 3$, $r_1 = 1, r_2 = 3$ and $r_3 = 2$

(i) The first jump is defined by the number J_1 which normally corresponds to the first residue in X. Thus $J_1 = 1$. The first location is defined by: $(X - J_1)$
Therefore:

$$X - 1 = \begin{bmatrix} |1-1|_7 = 0 \\ |3-1|_4 = 2 \\ |2-1|_3 = 1 \end{bmatrix}$$

(ii) The second jump is defined by the number J_2, such that:

$$J_2 = 7K_2 \text{ and } |2 - J_2|_4 = 0$$

$$|2 - 7K_2|_4 = 0 \Longrightarrow K_2 = \left|\frac{2}{7}\right|_4 = 2$$

Thus $J_2 = 14$

The second location is defined by: $(X - J_1 - J_2)$.

Therefore:

$$X - 1 - 14 = \begin{bmatrix} |0-14|_7 = 0 \\ |2-14|_4 = 0 \\ |1-14|_3 = 2 \end{bmatrix}$$

(iii) The third jump is defined by the number J_3, such that:

$J_3 = 7 * 4K_3$ and $|2 - J_3|_3 = 0$

$|2 - 28K_3|_3 = 0 \Longrightarrow K_3 = \left|\frac{2}{28}\right|_3 = 2$

Thus $J_3 = 56$

The third location is defined by: $(X - J_1 - J_2 - J_3)$

Therefore:

$$X - 1 - 14 - 56 = \begin{bmatrix} |0-56|_7 = 0 \\ |0-56|_4 = 0 \\ |2-56|_3 = 0 \end{bmatrix}$$

Therefore the corresponding equivalent decimal number is: $J_1 + J_2 + J_3 = 1 + 14 + 56 = 71$

Thus, $(1, 3, 2)_{RNS} = 71_{decimal}$

5. PERFORMANCE EVALUATION

In this section, the performance of the proposed scheme is evaluated with similar state of the art scheme proposed in [2]. In the first place, the method proposed in this paper is applicable to both moduli sets with common factors and those without common factors. This is demonstrated clearly with the choice of the moduli set $\{2^{n+1} - 1, 2^n + 1, 2^n - 1\}$ which share common factors when n is odd. Traditional conversion methods built on the CRT and the MRC techniques normally will fail in such cases. However, the scheme proposed here is capable of completing the reverse conversion process when n is both even and odd. Also, the proposed scheme does not depend on the big M and multiplicative inverses which will normally slow down the speed of most CRT and MRC based converters.

Table 1*:* A comparison of complexities

Scheme	*DR*	*AREA*	*DELAY*
[2]	$4n+1$	$20n+5$	$12n+7$
Proposed	$3n+1$	$13n+2$	$7n+2$

From table1, it can be seen that the proposed scheme generally, has outperformed the scheme proposed in [2] in terms of area and delay. It can be noticed that whilst the proposed scheme has a dynamic range of $3n+1$, that of the scheme proposed in [2] has a dynamic range of $4n+1$. A relativity analysis also showed that the proposed scheme achieved a reduction in area of about 68% and 53% reduction in delay. This is shown clearly in Fig 2.

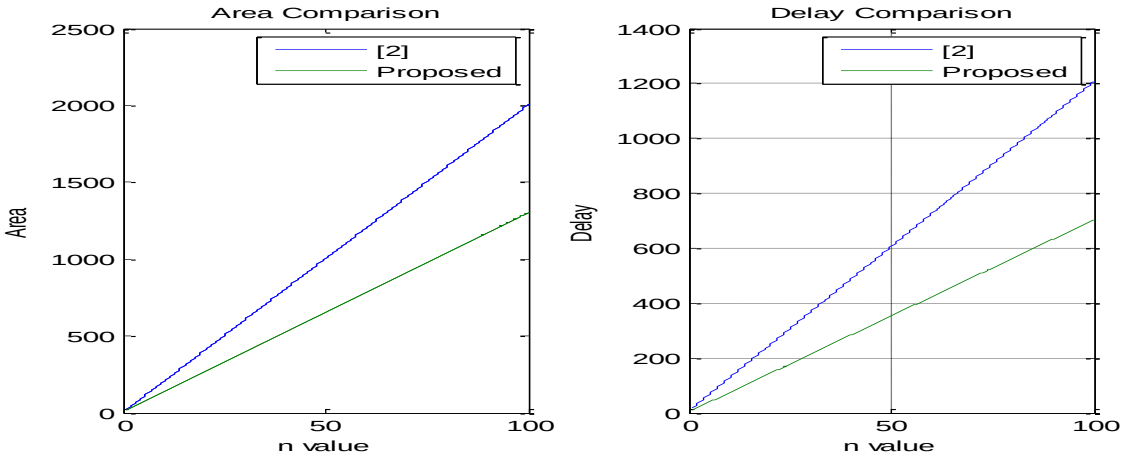

Fig 2: Graphs of the Area and Delay comparisons

6. CONCLUSION

In this paper, a hybrid method, involving the Cyclic Jump technique which encapsulates the Mixed Radix Conversion Method is presented. It is observed that the method accommodates moduli sets with common factors and those which are relatively prime. Also the proposed scheme is mainly based on modular computations and generates smaller numbers which makes computations easier and faster. It is further observed that parameters such as multiplicative inverse and big M which often increase the computational time of most existing techniques are absent in the proposed scheme. In the final analysis, the proposed scheme is seen to perform better at about 68% in terms of area and 53% in terms of delay when compared with the proposal in [2].

REFERENCES

[1] A. Omondi and B. Premkumar, *Residue Number Systems: Theory and Implementation*, vol. 2. Published by Imperial College Press and Distributed by World Scientific Publishing Co., 2007.

[2] P. A. Agbedemnab, M., A. Agebure, and S. Akobre, "A Fault Tolerant Scheme for Detecting Overflow in Residue Number Microprocessors," *Int. J. Eng. Comput. Sci.*, vol. 7, no. 2, pp. 23578–23584, Feb. 2018.

[3] P. A. Agbedemnab and E. K. Bankas, "A Novel RNS Overflow Detection and Correction Algorithm for the Moduli Set {2^n-1,2^n,2^n+1}," *Int. J. Comput. Appl.*, vol. 110, no. 16, pp. 30–34, Jan. 2015.

[4] S. J. Piestrak, "A high-speed realization of a residue to binary number system converter," *IEEE Trans. Circuits Syst. II Analog Digit. Signal Process.*, pp. 661–663, Oct. 1995.

[5] M. I. Daabo, "Overflow Detection Schemes for Residue Number System Architecture," PhD Thesis, University for Development Studies, Tamale, Ghana, 2015.

[6] R. C. Debnath and D. A. Pucknell, "On multiplicative overflow detection in residue number system," *Electron. Lett.*, vol. 14, no. 5, pp. 129–130, Mar. 1978.

[7] H. Yassine M., "Fast Arithmetic Based on Residue Number System Architectures," in *IEEE ISCAS'91*, Singapore, 1991, pp. 2947–2950.

[8] M. Askarzadeh, M. Hosseinzadeh, and K. Navi, "A New Approach to Overflow Detection in Moduli Set 2n-3, 2n-1, 2n+1, 2n+3," in *Second International Conference on Computer and Electrical Engineering, 2009. ICCEE '09*, 2009, vol. 1, pp. 439–442.

[9] M. Rouhifar, M. Hosseinzadeh, S. Bahanfar, and M. Teshnehlab, "Fast Overflow Detection in Moduli set {2^n-1,2^n,2^n+1}," *Int. J. Comput. Sci. Issues*, vol. (8/3), pp. 407–414, May 2011.

[10] D. Younes and P. Steffan, "Universal Approaches for Overflow and Sign Detection in Residue Number System Based on {2n – 1, 2n, 2n + 1}," presented at the ICONS 2013, The Eighth International Conference on Systems, 2013, pp. 77–81.

[11] H. Siewobr and K. A. Gbolagade, "RNS Overflow Detection by Operands Examination," *Int. J. Comput. Appl.*, vol. 85, no. 18, pp. 1–5, Jan. 2014.

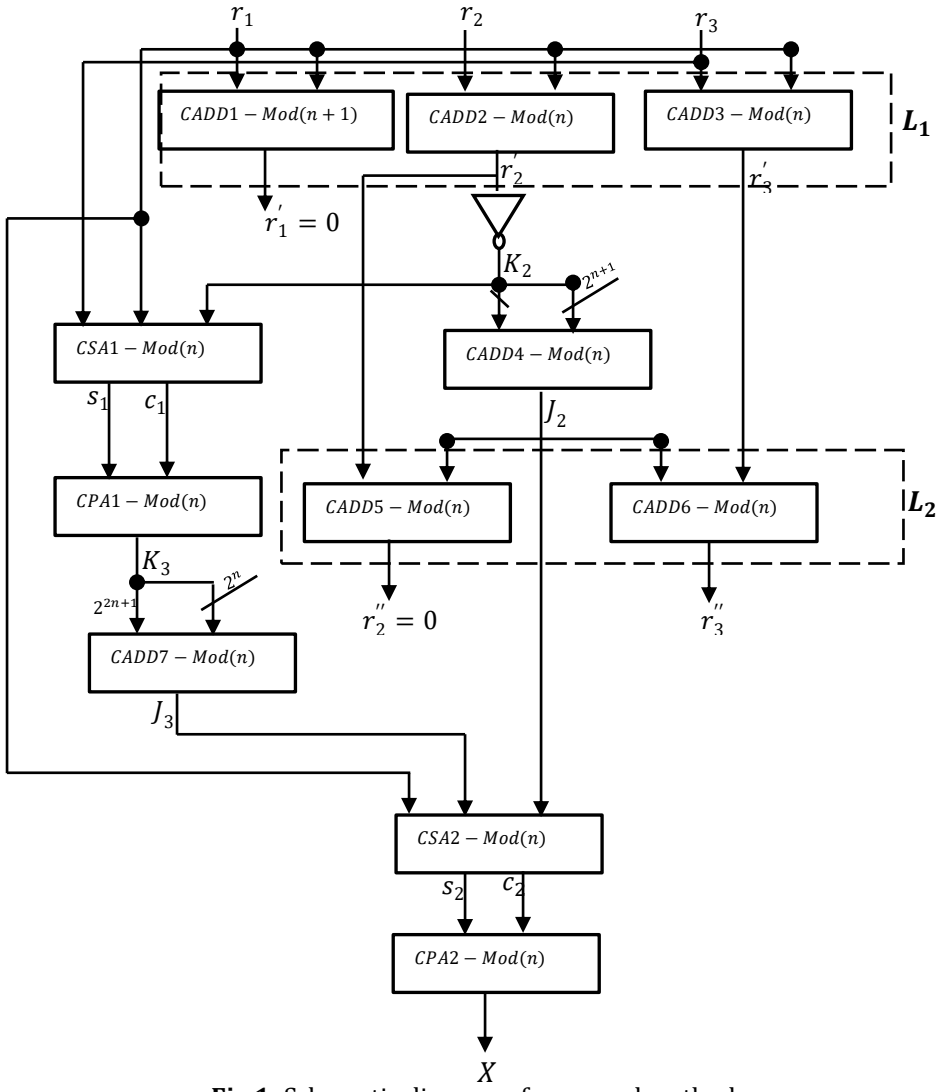

Fig 1: Schematic diagram of proposed method