Novel Parallel - Perfix Structure Binary to Residue Number System Conversion Method

Omar Dajani
Pepe Siy Wayne State Univerisity
Department of Electrical and Computer Engineering

Abstract

In the present world there is always a demand for faster algorithms and techniques that could boost up the speed of the computations. With the help of VLSI fabrication techniques and using residue number system (RNS) arithmetic we can achieve the faster speeds. In this paper we propose a novel parallel prefix binary to residue number system conversion method. The method that we present in this paper utilizes parallel-prefix technique with multiplexers and modulo adders as the main building blocks which makes it practical and suitable for VLSI implementation.

Keywords: Residue Number System, Binary to Residue Conversion, Multiplexer, Modulo Adder.

1. INTRODUCTION

In the last decade, Residue number system (RNS) has received increased attention due to its ability to support high speed concurrent arithmetic applications [1-3] such as Fast Fourier transform (FFT), image processing and digital filters utilize the efficiencies of RNS arithmetic in addition and multiplication. The advancements in very large scale integration (VLSI) technology and demand for parallelism computation have enabled researchers to consider RNS as an alternative approach to high speed concurrent arithmetic.

Several methods are found in literature for binary to RNS conversion. Alia and Martinelli [4] have proposed a method for binary to residue conversion based on powers of 2. A modification to the above method was proposed by Cappocelli and Giancarlo [5]. Anandmohan [6] has proposed a similar method but with difference that his method is based on the cyclic property of power of 2 moduli set. Behrooa[7] proposed a table lookup schemes for binary to Residue conversions.

In this paper, we present a novel binary to Residue Number System conversion method that we used to build Residue Arithmatic logic unit (RALU). RALU has three main units: Binary to Residue unit, ALU and Residue to Binary unit [8]. The organization of this paper is as follows. Section two explains RNS system. In section three we present new conversion from binary to RNS algorithm. Section four and five show illustrative example and implementation selection techniques. Section six is comparison between the new method and pervious work. Conclusion is in section seven.

2. RESIDUE NUMBER SYSTEM

Any n-bit nonnegative integer number X, in the range $0 \le X \le 2^n-1$ is represented in binary number system as $X = 2^{n-1} b_{n-1} + ... + 2^2 b_2 + 2 b_1 + b_0 = \sum_{j=0}^{n-1} 2^j b_j$

where $b_j \in \{0, 1\}$.

Meanwhile in RNS, X is represented by k residue digits x_i as $X = \{x_1, x_2, x_3, \ldots, x_k\}$ where $x_i = X \bmod m_i$ and m_i belong to set of relatively prime moduli; $m_i \in \{m_1, m_2, m_3, ..., m_k\}$ [9]. If

the moduli are relatively prime numbers, there is a unique RNS representation for each integer in range $0 \le X \le \prod_{i=1}^{s} m_i$

3. NEW NOVEL CONVERSION METHOD FROM BINARY TO RESIDUE REPRESENTATION

As shown above an integer number X can be represented in Binary system as

$$X = 2^{n-1} b_{n-1} + ... + 2^2 b_2 + 2 b_1 + b_0 = \sum_{j=0}^{n-1} 2^j b_j$$

And RNS representation of number X is

$$|X|_m = \left|\sum_{j=0}^{n-1} \left|2^j b_j\right|_m\right|_m \qquad \text{for } m > 2 \qquad \text{for } m > 2$$

$$= \left|\sum_{j=0}^{n-1} \left|2^j\right|_m b_j\right|_m \qquad (1)$$

Let $M_{A_1A_0} = (A_1, A_0)[Y_0, Y_1, Y_2, Y_3]$ denotes a 2-bit multiplexer where the 2 control bits (A1, A0) select the inputs (Y_0, Y_1, Y_2, Y_3) to be outputted

Lemma 1: For any pair of bits b_j& b_i for j & i ≥ 0,

$$\left|\left|2^j\right|_m b_j + \left|2^i\right|_m b_i\right|_m = \left|X_{ji}\right|_m$$

can be implemented using 2-bit multiplexer :

$$M_{ji} = (b_j, b_i)[0, \left|2^i\right|_m, \left|2^j\right|_m, \left|\left|2^j\right|_m + \left|2^i\right|_m\right|_m] \qquad (2)$$

Where the control bits (A_1, A_0) equal (b_j, b_i)

Proof:

Rewrite equation $\left|\left|2^j\right|_m b_j + \left|2^i\right|_m b_i\right|_m$ as

$$(0.\bar{b}_j.\bar{b}_i) + (\left|2^i\right|_m.\bar{b}_j.b_i) + (\left|2^j\right|_m.b_j.\bar{b}_i) + (\left|\left|2^j\right|_m + \left|2^i\right|_m\right|_m.b_j.b_i)$$

This is equivalent to 2-bit multiplexer M_{ji} with control bits (A_1, A_0) equal (b_j, b_i). Figure (1) shows the implementation for equation (2) with $b_j = b_1$ and $b_i = b_0$

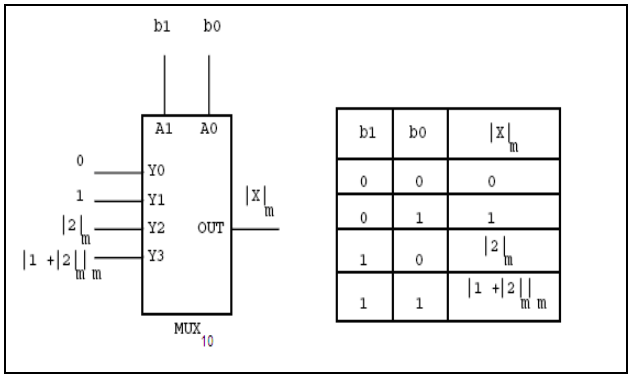

FIGURE 1: Two Bits (b1 & b0) Binary to Residue Number System Conversion

This pre-processing operator M_{ji} is represented in acyclic graph as node " ○ " in figure (2a), where all the inputs are constants and pre-calculated .

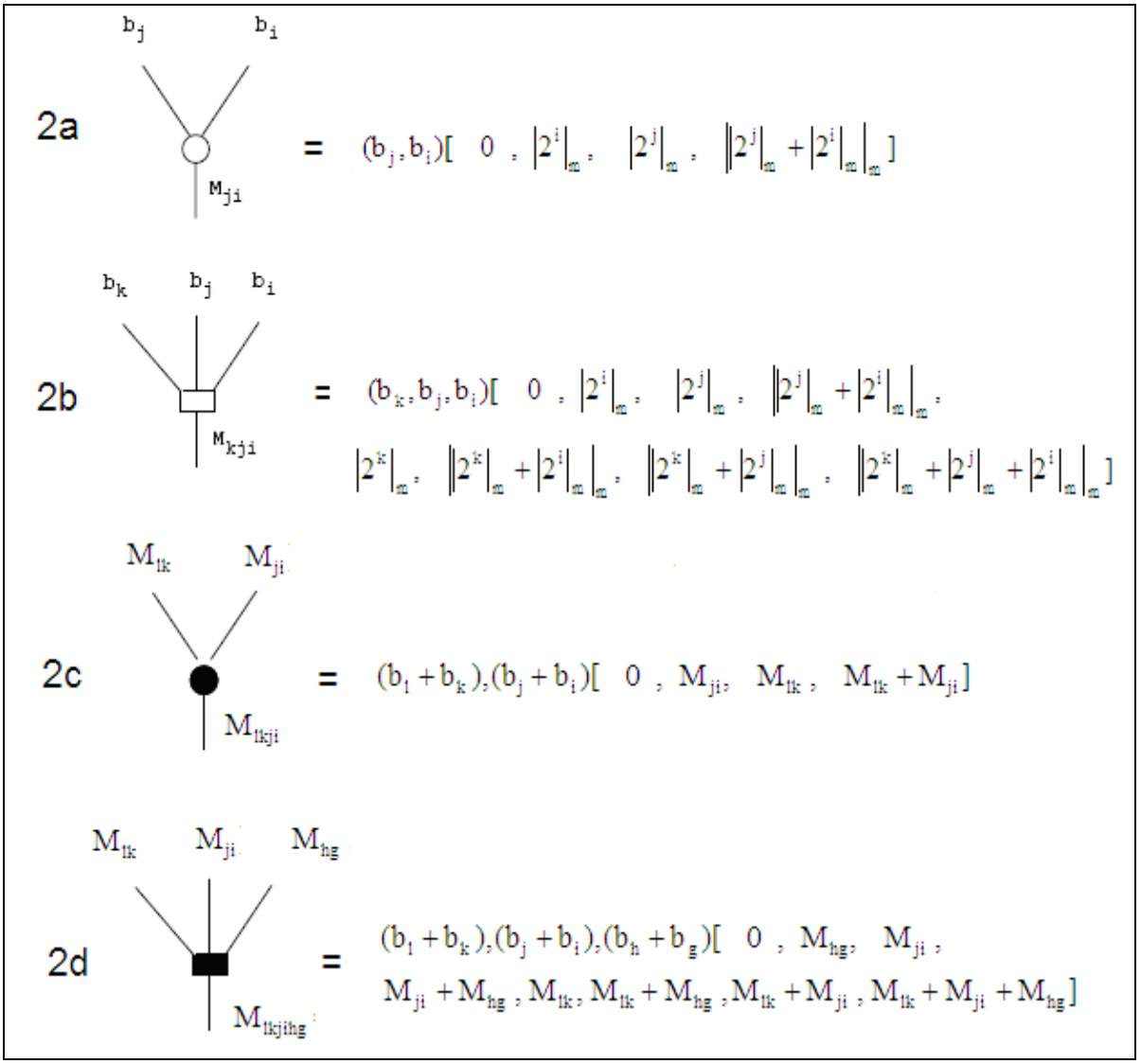

FIGURE 2: Prefix logic operation and their implementation

In three bit system, let $M_{A_2A_1A_0} = (A_2, A_1, A_0)[Y_0,\ \ Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, Y_7]$ denotes a 3-bit multiplexer where the 3 control bits (A_2, A_1, A_0) select the inputs ($Y_0, Y_1, Y_2, Y_3, Y_4, Y_5, Y_6, Y_7$) to be outputted.

Lemma 2: For any three bits b_k, b_j & b_i for k, j & i ≥ 0,
$\left\| 2^k \right|_m b_k + \left| 2^j \right|_m b_j + \left| 2^i \right|_m b_i \Big|_m = \left| X_{kji} \right|_m$ can be
implemented using 3-bit multiplexer :

$$M_{kji} = (b_k, b_j, b_i)[0\ , \left| 2^i \right|_m, \left| 2^j \right|_m, \left\| 2^j \right|_m + \left| 2^i \right|_m |_m, \left| 2^k \right|_m, \left\| 2^k \right|_m + \left| 2^i \right|_m |_m,$$
$$\left\| 2^k \right|_m + \left| 2^j \right|_m |_m, \left\| 2^k \right|_m + \left| 2^j \right|_m + \left| 2^i \right|_m |_m] \qquad (3)$$

Where control bits (A_2, A_1, A_0) equal (b_k, b_j, b_i)

Proof:

Rewrite $\left|\left|2^k\right|_m b_k + \left|2^j\right|_m b_j + \left|2^i\right|_m b_i\right|_m$ as

$$(0.\bar{b}_k.\bar{b}_j.\bar{b}_i) + (\left|2^i\right|_m.\bar{b}_k\bar{b}_j.b_i) + (\left|2^j\right|_m.\bar{b}_k.b_j.\bar{b}_i) +$$
$$(\left|\left|2^j\right|_m + \left|2^i\right|_m\right|_m.\bar{b}_k.b_j.b_i) + (\left|2^k\right|_m.b_k.\bar{b}_j.\bar{b}_i) +$$
$$(\left|\left|2^k\right|_m + \left|2^i\right|_m\right|_m, b_k\bar{b}_j.b_i) + (\left|\left|2^k\right|_m + \left|2^j\right|_m\right|_m b_k.b_j.\bar{b}_i) +$$
$$(\left|2^k\right|_m + \left|2^j\right|_m + \left|2^i\right|_m.b_k.b_j.b_i)$$

Above equation is equivalent to 3-bit multiplexer with b_k, b_j & b_i as selection control inputs. Figure (3) shows the implementation for equation 3 with $b_k = b_2$ $b_j = b_1$ and $b_i = b_0$

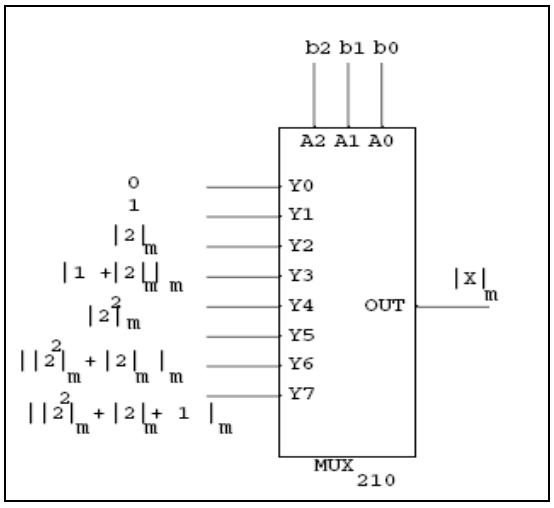

FIGURE 3: Three bits (b_2, b_1, b_0) Binary to Residue Number System Conversion

This pre-processing operator M_{kji} is represented in acyclic graph as node " □ " in figure (2b), where all the inputs are constants and pre-calculated.

Theorem 1: For Any two pairs of bits (b_l & b_k) (b_j & b_i for j, i , l & k ≥ 0 with the given expression

$$\left|\left|2^l\right|_m b_l + \left|2^k\right|_m b_k + \left|2^j\right|_m b_j + \left|2^i\right|_m b_i\right|_m = \left|X_{lkji}\right|_m$$

can be implemented using 2-bit multiplexer

$$M_{lkji} = (b_l + b_k),(b_j + b_i)[0, M_{ji}, M_{lk}, M_{lk} + M_{ji}] \qquad (4)$$

Where control bits (A_1, A_0) equal (b_l,+b_k , b_j,+b_i)

Proof $\left|X_{lkji}\right|_m = \left|\left|2^l\right|_m b_l + \left|2^k\right|_m b_k + \left|2^j\right|_m b_j + \left|2^i\right|_m b_i\right|_m$

$$\left|X_{lkji}\right|_m = \left|M_{lk} + M_{ji}\right|_m \qquad (5)$$

Where $M_{lk} = \left|\left|2^l\right|_m b_l + \left|2^k\right|_m b_k\right|_m$ and $M_{ji} = \left|\left|2^j\right|_m b_j + \left|2^i\right|_m b_i\right|_m$ by Lemma 1

Let $b_{lk} = (b_l + b_k)$ and $b_{ji} = (b_j + b_i)$

Rewrite equation (5) as $(0.\bar{b}_{lk}.\bar{b}_{ji}) + (M_{ji}.\bar{b}_{lk}.b_{ji}) + (M_{lk}.b_{lk}.\bar{b}_{ji}) + (M_{lk} + M_{ji}.b_{lk}.b_{ji})$.

And this is equivalent to 2-bit multiplexer M_{lkji} with control bits (A_1, A_0) equal (b_l+ b_k , b_j+ b_i). Figure (4) shows implementation for two pair bits (b_3, b_2) & (b_1, b_0)

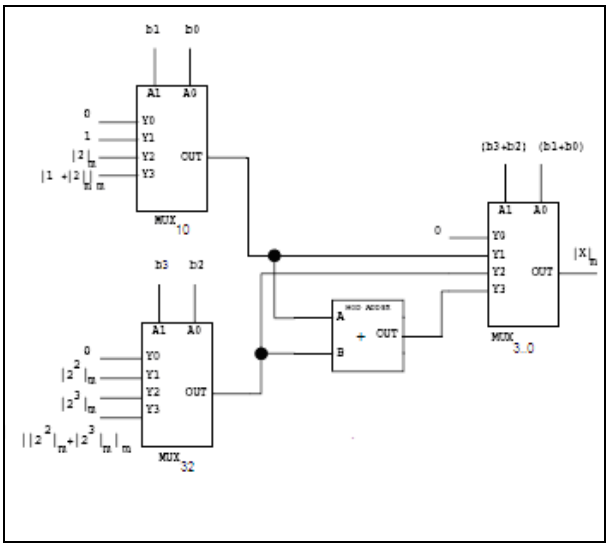

FIGURE 4: Four Bits Binary to RNS $\left|X_{3..0}\right|_m = \left|\left|2^3\right|_m b_3 + \left|2^2\right|_m b_2 + \left|2^1\right|_m b_1 + b_0\right|_m$

Lemma 3:

Combining two pairs of bits (b_l & b_k) (b_j & b_i) requires one 2-bit multiplexer and one 2 input mod adder. The delay time $\tau_{Total} = \tau_{mux_2} + \tau_{modadder_2}$

Proof

Equation (4) and figure (4) show that $(b_l + b_k),(b_j + b_i)[0, M_{ji}, M_{lk}, M_{lk} + M_{ji}]$ is equivalent to one 2-bit multiplexer and one 2-input mod adder; and delay time is equal to $\tau_{mux_2} + \tau_{modadder_2}$.

Figure (2c) represents acyclic graph " ● " for node M_{lkji} where M_{lk} & M_{ji} are inputs.

Lemma 4

The parallel prefix operator ● has the following properties

1) Commutative

$$M_{lk} \bullet M_{ji} = M_{ji} \bullet M_{lk}$$

2) Associative

$$M_{lk} \bullet (M_{hg} \bullet M_{ji}) = (M_{lk} \bullet M_{hg}) \bullet M_{ji}$$

Proof:

$$M_{lk} \bullet M_{ji} = M_{lkji}$$

$$= (b_l + b_k),(b_j + b_i)[0, M_{ji}, M_{lk}, M_{lk} + M_{ji}]$$

$$= \left|\left|2^l\right|_m b_l + \left|2^k\right|_m b_k + \left|2^j\right|_m b_j + \left|2^i\right|_m b_i\right|_m \quad (6)$$

$$M_{ji} \bullet M_{lk} = M_{jilk}$$

$$= (b_j + b_i),(b_l + b_k)[0, M_{kl}, M_{ji}, M_{ji} + M_{lk}]$$

$$= \left|\left|2^j\right|_m b_j + \left|2^i\right|_m b_i + \left|2^l\right|_m b_l + \left|2^k\right|_m b_k\right|_m \quad (7)$$

Both expressions (6) and (7) are the same by commutative property of "+" hence ● operator is commutative

$$M_{lk} \bullet (M_{hg} \bullet M_{ji}) = M_{lk} \bullet M_{hgji}$$
$$= M_{lkhgji}$$
$$= \left\| 2^{l} \right|_m b_k + \left| 2^{k} \right|_m b_k + \left| 2^{h} \right|_m b_h + \left| 2^{g} \right|_m b_g + \left| 2^{j} \right|_m b_j + \left| 2^{i} \right|_m b_i \Big|_m \quad (8)$$

$$(M_{lk} \bullet M_{hg}) \bullet M_{ji} = M_{lkhg} \bullet M_{ji}$$
$$= M_{lkhgji}$$
$$= \left\| 2^{l} \right|_m b_k + \left| 2^{k} \right|_m b_k + \left| 2^{h} \right|_m b_h + \left| 2^{g} \right|_m b_g + \left| 2^{j} \right|_m b_j + \left| 2^{i} \right|_m b_i \Big|_m \quad (9)$$

Both expressions (8) and (9) are the same by associative property of "+" hence ● operator is associative

Theorem 2: For any three pairs of bits (b_l & b_k) , (b_j & b_i) and (b_h & b_g) for l, k, j, i, h & g ≥ 0 with given expression

$$\left\| 2^{l} \right|_m b_l + \left| 2^{k} \right|_m b_k + \left| 2^{j} \right|_m b_j + \left| 2^{i} \right|_m b_i + \left| 2^{h} \right|_m b_h + \left| 2^{g} \right|_m b_g \Big|_m = \left| X_{lkjihg} \right|_m$$

can be implemented using 3-bit multiplexer

$$M_{lkjihg} = (b_l + b_k), (b_j + b_i), (b_h + b_g)[0, M_{hg}, M_{ji}, M_{ji} + M_{hg}, M_{lk}, M_{lk} + M_{hg},$$
$$M_{lk} + M_{ji}, M_{lk} + M_{ji} + M_{hg}] \quad (10)$$

Where control bits (A_2, A_1, A_0) equal (b_l+b_k , b_j+b_i , b_h+b_g,)

Proof:

$$\left| X_{lkjihg} \right|_m = \left\| 2^{l} \right|_m b_l + \left| 2^{k} \right|_m b_k + \left| 2^{j} \right|_m b_j + \left| 2^{i} \right|_m b_i + \left| 2^{h} \right|_m b_h + \left| 2^{g} \right|_m b_g \Big|_m$$

$$\left| X_{lkjihg} \right|_m = \left| M_{lk} + M_{ji} + M_{hg} \right|_m \quad (11)$$

Where $M_{lk} = \left\| 2^{l} \right|_m b_l + \left| 2^{k} \right|_m b_k \Big|_m$,

$M_{ji} = \left\| 2^{j} \right|_m b_j + \left| 2^{i} \right|_m b_i \Big|_m$ and

$M_{hg} = \left\| 2^{h} \right|_m b_h + \left| 2^{g} \right|_m b_g \Big|_m$ by Lemma 1

Let $b_{lk} = (b_l + b_k)$, $b_{ji} = (b_j + b_i)$ and $b_{hg} = (b_h + b_g)$

Rewrite equation (11) as

$$(0.\bar{b}_{lk}.\bar{b}_{ji}.\bar{b}_{hg}) + (M_{hg}.\bar{b}_{lk}\bar{b}_{ji}.b_{hg}) + (M_{ji}.\bar{b}_{lk}.b_{ji}.\bar{b}_{hg}) + ((M_{ji} + M_{hg})\bar{b}_{lk}.b_{ji}.b_{hg}) +$$
$$(M_{lk}.b_{lk}.\bar{b}_{ji}.\bar{b}_{hg}) + ((M_{lk} + M_{hg}).b_{lk}\bar{b}_{ji}.b_{hg}) +$$
$$((M_{lk} + M_{ji}).b_{lk}.b_{ji}.\bar{b}_{hg}) +$$
$$((M_{lk} + M_{ji} + M_{hg})b_{lk}.b_{ji}.b_{hg})$$

This is equivalent to 3-bit multiplexer M_{lkjihg} with control bits (A_2, A_1, A_0) equal (b_l+ b_k , b_j+ b_i , b_h+ b_g)

Figure (5) shows implementation for three pairs of bits (b_5, b_4), (b_3, b_2) & (b_1, b_0)

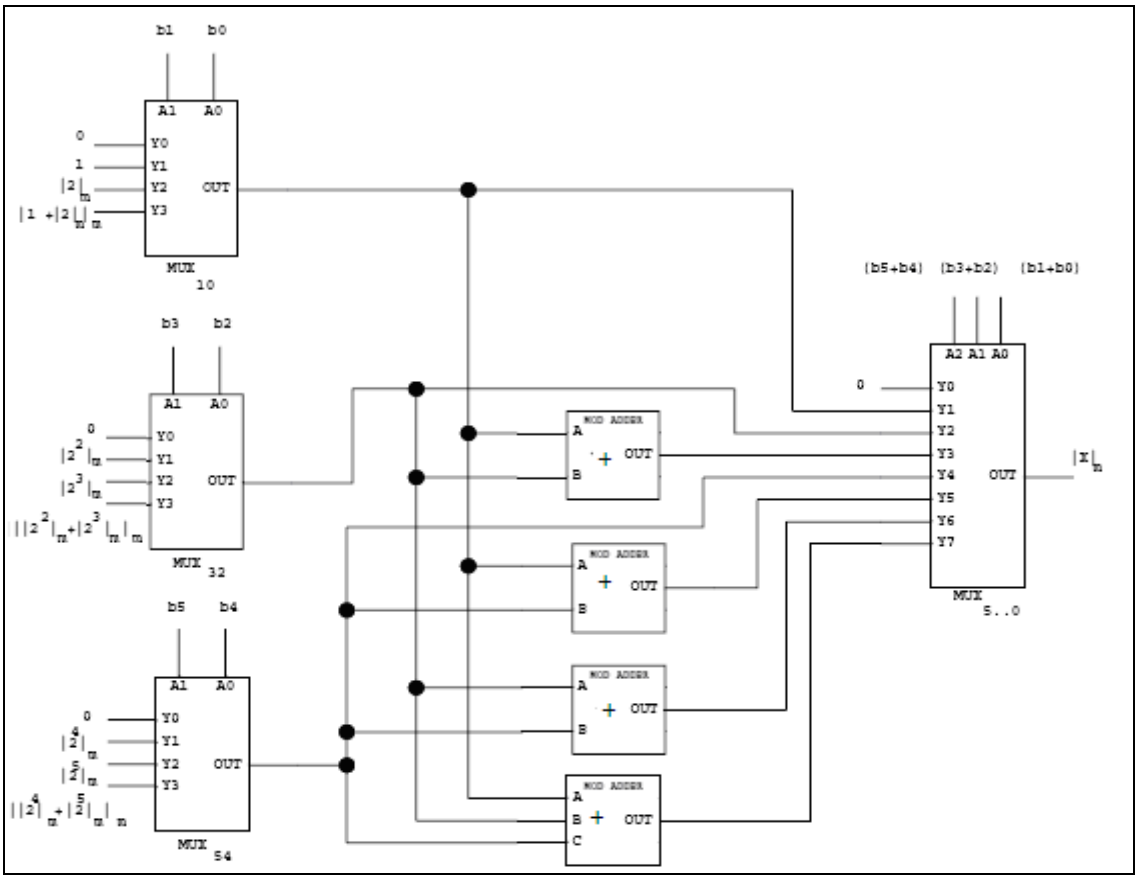

FIGURE 5: Six Bits Binary to Residue Number System Conversion

$$\left|X_{5..0}\right|_m = \left|\left|2^5\right|_m b_5 + \left|2^4\right|_m b_4 + \left|2^3\right|_m b_3 + \left|2^2\right|_m b_2 + \left|2^1\right|_m b_1 + b_0\right|_m$$

Lemma 5:

Combining three pairs of bits (b_l & b_k), (b_j & b_i) & (b_h & b_g) requires one 3- bit multiplexer and three 2-input mod adder and one 3-input mod adder. The delay time equals

$$\tau_{Total} = \tau_{mux_3} + 2\tau_{modadder_2} = \tau_{mux_3} + \tau_{modadder_3}$$

Proof

Equation (10) and figure (5) show that $(b_l + b_k),(b_j + b_i),(b_h + b_g)[0, M_{hg},$ $M_{ji}, M_{ji} + M_{hg}, M_{lk}, M_{lk} + M_{hg}, M_{lk} + M_{ji}, M_{lk} + M_{ji} + M_{hg}]$ is equivalent to one 3-bit multiplexer and three 2-input mod adder and one 3-input mod adder; and delay time is equal to $\tau_{mux_3} + 2\tau_{modadder_2}$

Figure (2d) represents acyclic graph for "■ " for node M_{lkhgji} where $M_{lk,}$ M_{hg} & M_{ji} are inputs

Lemma 6:

The parallel prefix operator ■ has the following properties

1) Commutative

M_{lkjihg} ■ M_{tsrqpo} = M_{tsrqpo} ■ M_{lkjihg}

2) Associative

M_{lkjihg} ■ (M_{tsrqpo} ■ M_{zyxwru}) = (M_{lkjihg} ■ M_{tsrqpo}) ■ M_{zyxwru}

Proof:
The proof is similar to Lemma 4

4. ILLUSTRATIVE EXAMPLE

In this section, we will use illustrate how theorem 1, theorem 2, lemma 1 and lemma 2 can be combined to design a binary to residue convertor. Figure (6) shows how $|X|_m$ for n = 8 is computed.
In the first layer, using pre-processing operator ○ each consecutive pair of bits are group together (b_7, b_6) (b_5 , b_4) (b_3 , b_2) (b_1, b_0) creating nodes M_{76}, M_{54}, M_{32}, M_{10} . In the second layer, using parallel prefix operator ● each consecutive M node are combined (M_{76}, M_{54}) (M_{32}, M_{10}) forming nodes $M_{7..4}$ & $M_{3..0.}$ In the last layer, using parallel prefix operator ● the last 2 M nodes are combined ($M_{7..4}$, $M_{3..0}$) forming node $M_{7..0}$ = $|X_{7..0}|_m$. Figure (7a) shows the actual hardware implementation.

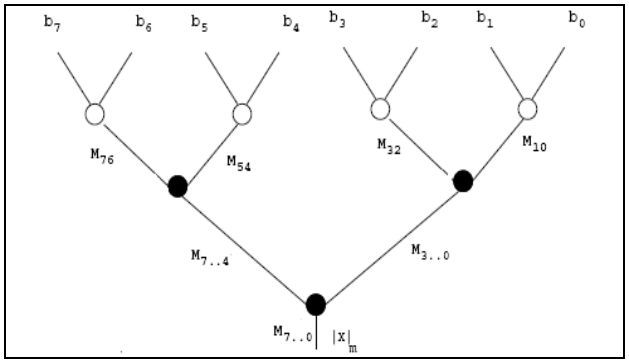

FIGURE 6: Prefix Structure of 8 Bits Binary to RNS

Total delay time for this example is calculated by counting the delay introduce by the operator in each layer
by using lemma 3 as follows

Layer 1: delay time is τ_{mux_2}

pre-processing operator ○ doesn't requires an adder

Layer 2 : delay time is $\tau_{mux_2} + \tau_{modadder_2}$

Layer 3: delay time is $\tau_{mux_2} + \tau_{modadder_2}$

Total delay is the sum of all layers delay time $\tau_{Total} = 3\tau_{mux_2} + 2\tau_{modadder_2}$

To show that hardware works, the signal propagation for binary number $|X|_7 = |(1110 \quad 0110)_2|_7 = |244_{10}|_7 = 6$ is illustrated in figure (7b). Similarly, the reader can try any bit pattern in figure (7b) to check the validity of the design. For example $|X|_7 = |(1111 \quad 1111)_2|_7 = |255_{10}|_7 = 3$ where each multiplexer select line is 3 and the selected output are shown in parenthesis.

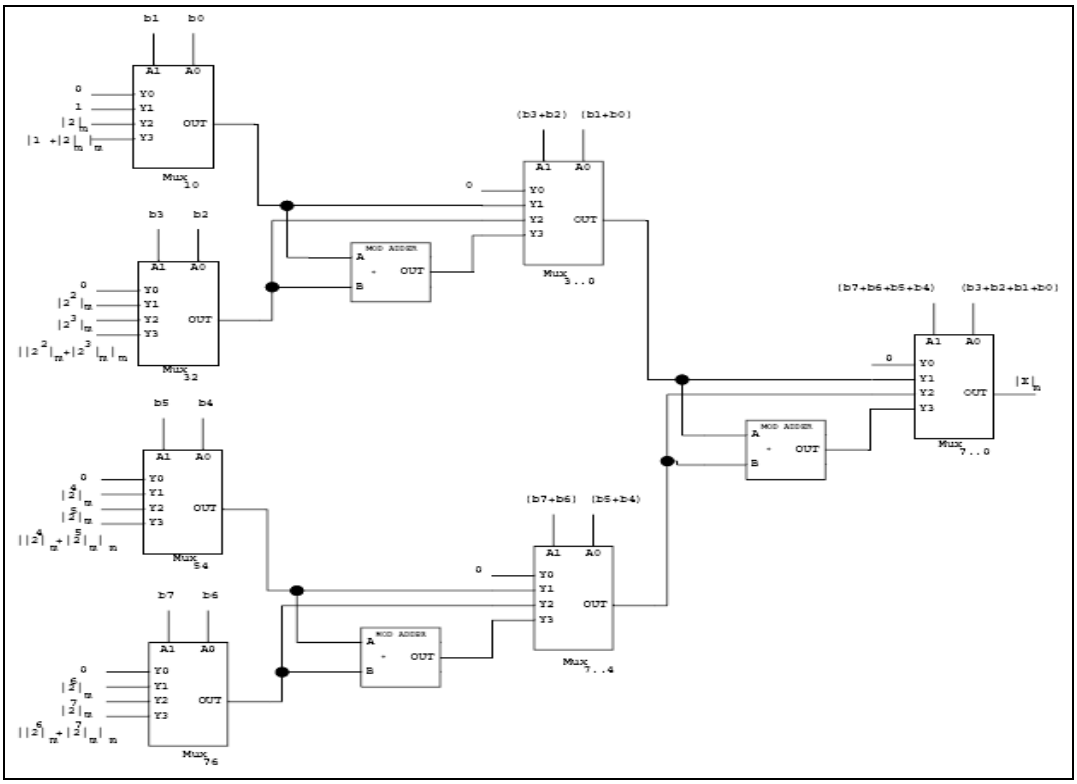

FIGURE 7A: Eight Bits Binary to Residue Number System Conversion

$$\left|X_{7..0}\right|_m = \left|\left|2^7\right|_m b_7 + \left|2^6\right|_m b_6 + \left|2^5\right|_m b_5 + \left|2^4\right|_m b_4 + \left|2^3\right|_m b_3 + \left|2^2\right|_m b_2 + \left|2\right|_m b_1 + b_0\right|_m$$

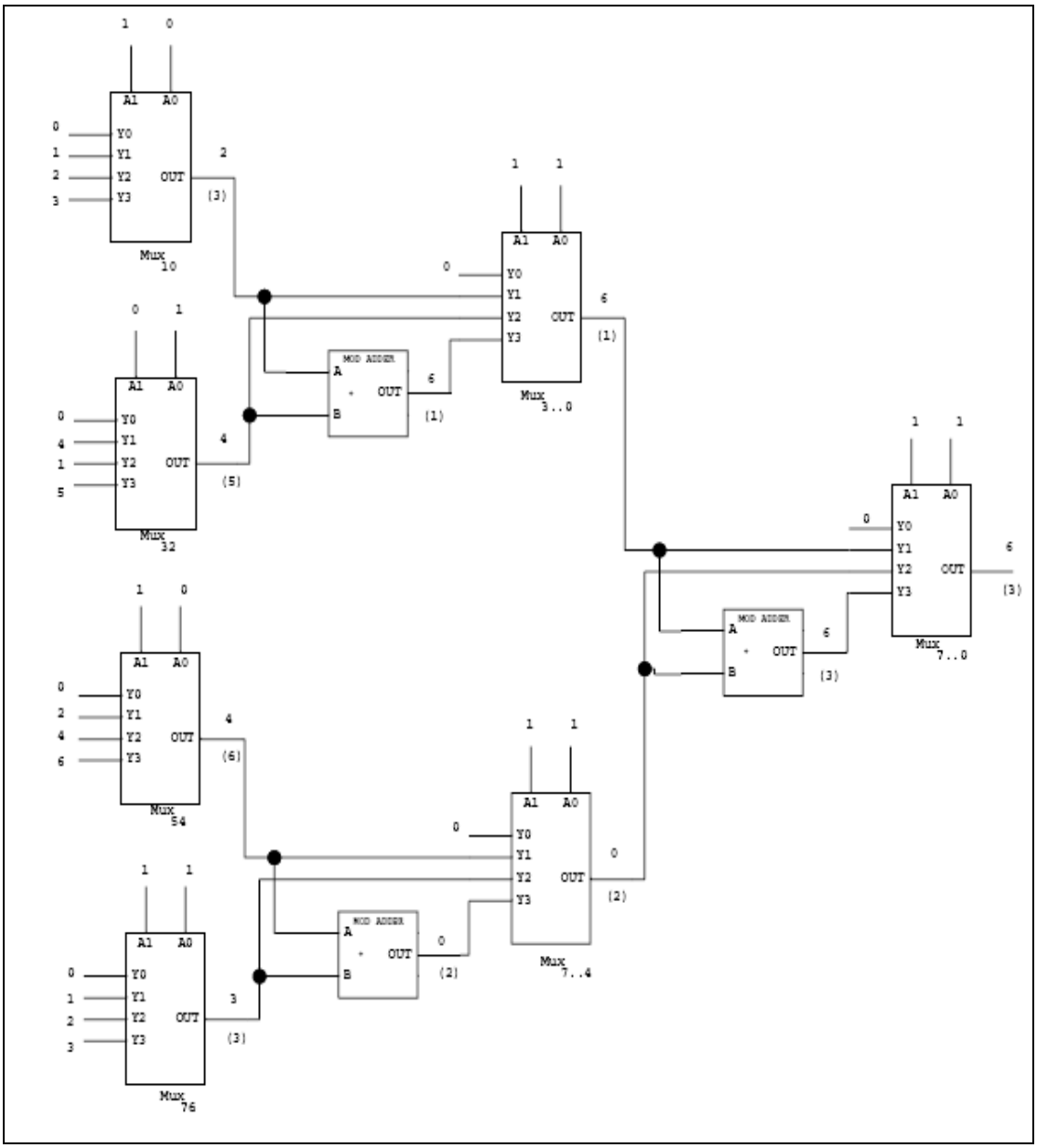

FIGURE 7B: Example for Signal propagation of $\left|(1110 \quad 0110)_2\right|_7$ and $\left|(1111 \quad 1111)_2\right|_7$

5. IMPLEMENTATION SELECTION

There are several possible binary to RNS imp- lementations using a combination of 2-bit and 3-bit multiplexers. Figure (8) shows three different imp- lementations (design 1, design 2 and design 3) for 10 bits binary to residue conversion system.

To simplify comparison, the following reasonable assumptions are made

$\tau_{\text{mux}_2} = \tau_{\text{mux}_3}$; $\tau_{\text{modadder}_3} = 2\tau_{\text{modadder}_2}$

Design 1 uses nine 2-bit multiplexers and four 2-input mod adders with

Layer 1: delay time is τ_{mux_2}

Layer 2: delay time is $\tau_{\text{mux}_2} + \tau_{\text{modadder}_2}$

Layer 3: delay time is $\tau_{mux_2} + \tau_{modadder_2}$

Layer 4: delay time is $\tau_{mux_2} + \tau_{modadder_2}$

Total delay is sum of all layers delay time $\tau_{Total} = 4\tau_{mux_2} + 3\tau_{modadder_2}$

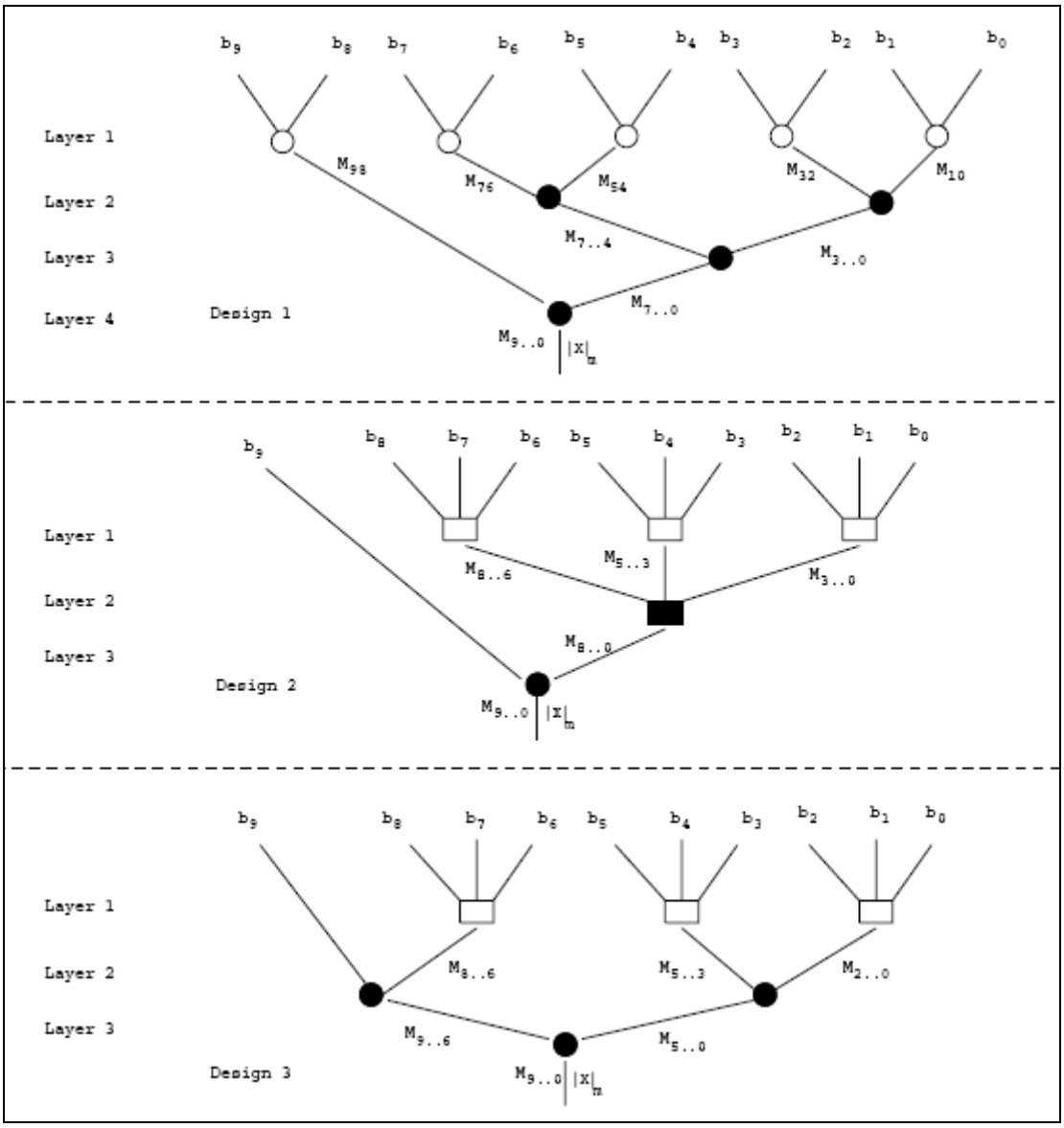

FIGURE 8: Prefix Structure of 10 Bits Binary to RNS

$$|X_{9..0}|_m = \left| |2^9|_m b_9 + |2^8|_m b_8 + |2^7|_m b_7 + |2^6|_m b_6 + |2^5|_m b_5 + |2^4|_m b_4 + |2^3|_m b_3 + |2^2|_m b_2 + |2|_m b_1 + b_0 \right|_m$$

Design 2 uses four 3-bit multiplexers, one 2-bit multiplexers, four 2-input mod adders and one 3-input adder with

Layer 1: delay time is τ_{mux_3}

Layer 2: delay time is $\tau_{mux_3} + 2\tau_{modadder_2}$

Layer 3: delay time is $\tau_{mux_2} + \tau_{modadder_2}$

$$\tau_{Total} = \tau_{mux_2} + 2\tau_{mux_3} + 3\tau_{modadder_2}$$
$$= 3\tau_{mux_2} + 3\tau_{modadder_2}$$

Design 3 uses three 3-bit multiplexers, three 2-bit multiplexer and three 2-input mod adders with

Layer 1: delay time is τ_{mux_3}

Layer 2 : delay time is $\tau_{mux_2} + \tau_{modadder_2}$

Layer 3: delay time is $\tau_{mux_2} + \tau_{modadder_2}$

$$\tau_{Total} = 2\tau_{mux_2} + \tau_{mux_3} + 2\tau_{modadder_2}$$
$$= 3\tau_{mux_2} + 2\tau_{modadder_2}$$

From Table (1), it shows that Design 3 uses less hardware and the fastest

Design		Hardware count			
#	Time Delay	Mux_2	Mux_3	Mod add_2	Mod add_3
1	$4\tau_{mux_2} + 3\tau_{modadder_2}$	9	0	4	0
2	$3\tau_{mux_2} + 3\tau_{modadder_2}$	1	4	4	1
3	$3\tau_{mux_2} + 2\tau_{modadder_2}$	3	3	3	0

TABLE 1: shows comparison between the three designs implementation for n = 10

6. COMPARISON SELECTION TO PERVIOUS WORK

This Novel method has hardware advantages than any competitive converters. In 1984, Alia and Martinelli [3] published binary to RNS conversion design based on power 2 mod m_i . The design uses processing elements (PE) and each PE is associated with two registers. Each of these registers is serially loaded with $\left|2^i\right|_m$ and $\left|2^{i+1}\right|_m$ respectively and it enabled to put their content or zeros' out depending on value 1 or 0 of b_j and b_{j+1} respectively. The two outputs are added in a modular adder. Thus, at first level, n/2 PEs are required. The number of stages in this method is $[\log_2 n]$ and after successive transformation and addition, the residue result is available. Cappocelli and Giancarlo [4] suggested the use of t PEs where $t = n/\log_2 n$, each PE computing the residue corresponding to k- bit binary word where $k = \log_2 n$, the residue 2^{kt} mod m_i is serially fed to $\hat{k}$th PE ($\hat{k} = 0, 1, 2, \ldots, t-1$), Based on these initial residues, the residues corresponding to the next (k-1) powers are computed by first doubling and then weighting according to the input bits in each PE. The partial residues of k-bit words computed over parallel t PEs are then added to yield the final residue. Anandmohan [5] has proposed a similar method but with a difference that X is divided into t sections based on the cyclic property of 2^j mod m_i. Using the fact that, 2^j, 2^{j+lo} and 2^{j+2lo} have the same residues due to periodicity of period *lo* , *lo* bits are first added. The width of the result is confined to *lo* bits by adding the carry bit resulting from previous addition to LSB of the result. The residue results is then determined by using methods given in [3]

7. CONCLUSIONS

In this paper we presented a new novel binary to Residue conversion method that eliminates the need for processing elements (PE) as the above competitive converter designs and doesn't use table lookup as in Behrooa Parhami [6]. The new method that we present here is based on multiplexers concept which makes it practical and suitable for VLSI implementation

8. REFERENCES

[1] M.A. Bayoumi, "Digital filter VLSI systolic arrays over finite fields for DSP applications," in Proc. 6th IEEE annual Phoenix Conf. Computers and Communications, pp 194-199, Feb 1987.

[2] M. A. Soderstrand et al., Eds., "Residue Number System Arithmetic: Modern Applications in Digital Signal Processing" New York: IEEE Press, 1986

[3] K. Konstantinides and V. Bhaskaran, “Monolithic architectures for image processing and compression,”IEEE Computer Graphics & Applications, pp. 75-86, Nov. 1992

[4] G. Alia and E. Martinelli, "A VLSI algorithm for direct and reverse conversion from weighted binary number to residue number system," IEEE Trans. Circuits Syst., vol. 31, pp. 1425–1431, Dec. 1984.

[5] R. M. Capocelli and R. Giancarlo, "Efficient VLSI networks for converting an integer from binary system to residue number system and vice versa," IEEE Trans. Circuits Syst., vol. 35, pp. 1425–1431, Nov. 1988.

[6] A. Mohan, “Novel design for binary to RNS converters,” in Proc. Int.Symp. Circuits and Systems, London, U.K., 1994, pp. 357–360.

[7] Behrooa Parhami, "Optimal Table-Lookup Schemes for Binary-to-Residue and Residue-to-Binary Conversions," IEEE Trans, Circuits Syst., 1993

[8] Mohamed. Akkal and Pepe Siy, "A new Mixed Radix Conversion algorithm", Journal of Systems Architecture, Volume 5, Iusse 9, September 2007, Pages 577-586

[9] N. S. Szabo and R. I. Tanaka, " Residue Arithmetic and Its Applications to Computer Technology". New York: McGraw Hill, 1967.