FIR Filter Implementation Based on the RNS with Diminished-1 Encoded Channel

Dragana Živaljević, Negovan Stamenković, and Vidosav Stojanović

***Abstract*—The nonrecursive digital filters implementation based on the RNS arithmetic is presented in this paper. In this implementation popular moduli set $\{2^n-1, 2^n, 2^n+1\}$ with diminished-one (diminished-1) 2^n+1 encoded channel is used. The diminished-one number system is used to avoid $(n+1)$-bit circuits in (2^n+1)-bit channel. Thus, in proposed approach all operand have n-bit length. The proposed RNS architecture of the filter consists of three main blocks: forward and reverse converter and arithmetic processor for each channel, where binary operations perform. Architecture for residue to binary (reverse) convertor with diminished-1 encoded channel and architecture for modulo multiplication have been proposed. Besides, for all RNS channels, the systolic design is used for the efficient realization of FIR filter. A numerical example illustrates the principles of diminished-1 residue arithmetic, signal processing, and decoding for FIR filters.**

***Keywords*—Chinese remainder theorem, diminished-one, FIR filters, residue number system, reverse converter.**

I. Introduction

In many digital signal processing systems, finite impulse response (FIR) digital filters are frequently used because of their stability and linear phase property. On other hand, they are not suitable for recent applications demanding real-time performance and low power consumption. The demand for real-time digital signal processing with respect to power consumption has forced the researchers to look for efficient arithmetic algorithms, which can implement high speed digital signal processor units. The systems based on Residue Number System (RNS) have become the most popular as they take advantage of all the benefits given by the parallelism and the carry free computations.

Filter realizations using Residue Number System (RNS) have been investigated in literature [1][2][3][4]. RNS is suitable for implementation of high-speed digital signal processing due to their inherent parallelism, modularity, fault toleration and local carry propagation properties. Arithmetic operations like multiplication and addition can be carried out more efficiently as RNS ensures localized carry propagation properties. RNS is particularly suitable for implementing FIR filters where multiplications and additions are the core operations. These features make RNS beneficial for digital signal processing applications, particularly, when large word length and high throughput rate are required.

D. Zivaljević, N. Stamenković and V. Stojanović are with University of Niš, Faculty of Electronic Engineering, A. Medvedeva 14, 18000 Niš, Serbia, e-mails: dragana.zivaljevic@elfak.ni.ac.rc; negovanstamenkovic@gmail.com; vidosav.stojanovic@elfak.ni.ac.rs.

Manuscript received February 27, 2012.

The selection of the moduli set plays a critical role in the improvement of the performance of RNS FIR filters [5]. The moduli set $\{2^n-1, 2^n, 2^n+1\}$ is used to design RNS FIR filters in this paper for the following reasons: firstly, it provides simpler design for converters; secondly, we can utilize the dedicated hardware multipliers on FPGA platforms to implement multiplications with acceptable cost for normally encoded RNS channel; thirdly, since it is the most commonly used one in previous works, using this moduli set makes our work applicable to the most existing designs.

The diminished-1 representation of binary numbers was introduced in [6] to speed up the modulo (2^n+1) arithmetic operations. Since only n bits are required for the representation of any digit in RNS, the diminished-1 representation can lead to implementations with delay and area that approach to delays and areas of the modulo (2^n-1) and 2^n representations. A lot of papers on the design of modulo (2^n-1) adders and multipliers for diminished-1 operands have already been published [7], [8], [9], [10]. However, special treatment is required for operands equal to zero. Since this can lead to implementations with increased area and delay complexity, the efficient integration of zero handling into modulo (2^n+1) arithmetic units is an open problem.

An RNS-based architecture of the filter consists of three main blocks. As first, all operands are converted into their corresponding sets of residues with binary-to-residue (forward) converters, according to the specified moduli set. Then, the arithmetic processing is performed in parallel in each channel following the corresponding modulo arithmetic. Finally, the RNS representation of the results is converted back to binary with residue-to-binary (reverse) converters.

Forward converters for RNS with diminished-1 encoded channel are proposed in [11], while modulo (2^n+1) adder architectures for diminished-1 operands with integrated zero handling are proposed in [12], and modulo (2^n+1) multipliers for diminished-1 operands including zero indication bits are proposed in [13].

This paper proposes a new approach to FIR filters design based on the RNS with diminished-1 encoded channel and also architecture for the reverse converter for RNS with diminished-1 encoded channel.

Organization of the paper is as follows; after recalling the diminished-1 arithmetic in Section II, the architecture of N-th order recursive digital filter is presented in Section III.

II. Diminished-1 Arithmetic

To represent all integers in RNS using modulo (2^n+1), $(n+1)$ bits are required. The additional bit is required in order to represent the number $2^n = \langle -1 \rangle_{2^n+1}$. To overcome the problem of performing binary arithmetic with this additional bit, a modified binary number system is used in order to avoid additions and multiplications involving the additional bit. This allows the additional bit to be only 1 when the number to be represented is 0, which can be achieved by subtracting 1 from the normal binary number.

The normal representation and this diminished-1 representation are indicated in the Table I for $n = 4$. When performing

TABLE I
CORRESPONDENCE BETWEEN NORMAL, BINARY AND DIMINISHED-1 REPRESENTATIONS

Normal	Binary	Diminished-1
0	00000	1
1	00001	2
2	00010	3
3	00011	4
4	00100	5
5	00101	6
6	00110	7
7	00111	8
8	01000	9 (-8)
9 (-8)	01001	10 (-7)
10 (-7)	01010	11 (-6)
11 (-6)	01011	12 (-5)
12 (-5)	01100	13 (-4)
13 (-4)	01101	14 (-3)
14 (-3)	01110	15 (-2)
15 (-2)	01111	16 (-1)
16 (-1)	10000	0

arithmetic for mod $(2^n + 1)$ using diminished-1 system, all input operands and the corresponding results are expressed in diminished-1 form. Let x' be diminished-1 representation of normal binary number $x \in [0, 2^n]$, namely

$$x' = \langle x - 1 \rangle_{2^n+1}. \tag{1}$$

In (1), when $x \neq 0$, $x' \in [0, 2^n - 1]$ is an n-bit number, therefore $(n + 1)$-bit circuits can be avoided in this case. However, when $x = 0$, $x' = 2^n$ is an $(n + 1)$-bit number. This leads to special treatment for $x' = 0$. According to this representation a number x' is represented as $x'_n X'$, where x'_n is the zero indication bit and $X' = x'_{n-1} x'_{n-2} \ldots x'_0$ is the magnitude representation [12].

A. Binary to diminished-1 RNS convertor

With the diminished-1 encoded channel, a very efficient binary to diminished-1 RNS converter is proposed for the $\{2^n - 1, 2^n, 2^n + 1\}$ moduli set [11]. An $3n$-bits integer $X \overset{RNS}{\longrightarrow} (x_1, x_2, x'_3)$ in the dynamic range is represented as

$$X = \sum_{i=0}^{3n-1} X_i 2^i = N_2 2^{2n} + N_1 2^n + N_0. \tag{2}$$

As explained above, the diminished-1 representation uses the modulo of $X - 1$ instead of X. Computation of the value x'_3 for this representation, with an identical approach as in standard $(2^n + 1)$ channel, is performed as:

$$x'_3 = \langle X - 1 \rangle_{2^n+1} = \langle 1 + N_2 + \overline{N}_1 + N_0 \rangle_{2^n+1}. \tag{3}$$

Only one CSA (carry save adder) and one full modulo $(2^n + 1)$ adder are required for the diminished-1 modulo $(2^n + 1)$ RNS converter [11].

B. Diminished-1 modulo $(2^n + 1)$ addition

Ordinary addition in diminished-1 number system is performed as follows [14]:

$$S' = \langle x' + y' + 1 \rangle_{2^n+1}. \tag{4}$$

Modulo $(2^n + 1)$ addition can be realized by an end-around-carry adder, where the carry-out is inverted and fedback into the carry-in, i.e. $c_{in} = \overline{c}_{out}$. This can be achieved with two adders to prevent a combinational loop. The carry-out inversion logic does not work when both summands are equal to zero. Therefore an additional AND gate has to be added as the control input for a multiplexer, which selects the correct output $x' + y' = 2^n$, when $x' = y' = 2^n$ [15].

Diminished-1 modulo $(2^n + 1)$ addition is now defined by:

$$\langle x' + y' + 1 \rangle_{2^n+1} = \begin{cases} 2^n & \text{if } x = 2^n \wedge y = 2^n \\ \langle x' + y' \rangle_{2^n+1} + 1 & \text{otherwise} \end{cases} \tag{5}$$

J.-L. Beuchat propose an alternative definition for modulo $(2^n + 1)$ addition [16]. Let us define the $(n + 2)$-bit integer $S = s_{n+1} s_n \ldots s_0 = x' + y'$. The modulo $(2^n + 1)$ addition can be expressed as:

$$\langle x' + y' + 1 \rangle_{2^n+1} = \langle x' + y' \rangle_{2^n+1} + s_{n+1} 2^n + \overline{s_{n+1} \vee s_n}. \tag{6}$$

Figure 1 depicts the resulting hardware operator which requires carry-propagate adder, a NOR gate and incrementer [16].

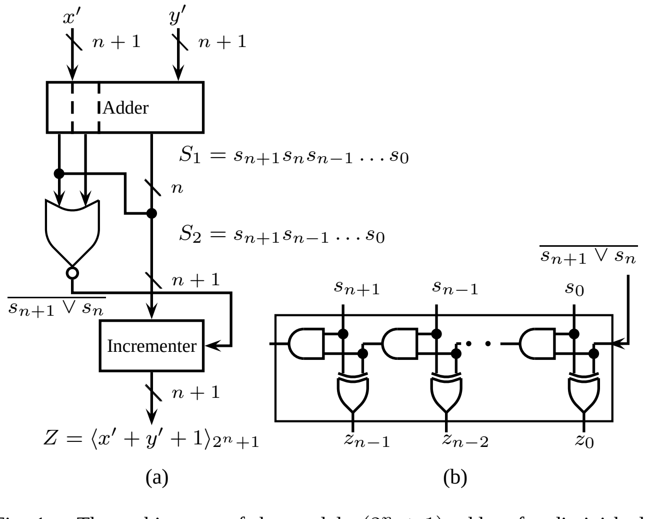
x'
y'
$n+1$
$n+1$
Adder
$S_1 = s_{n+1} s_n s_{n-1} \ldots s_0$
n
$S_2 = s_{n+1} s_{n-1} \ldots s_0$
$n+1$
$\overline{s_{n+1} \vee s_n}$
Incrementer
$n+1$
$Z = \langle x' + y' + 1 \rangle_{2^n+1}$
$\overline{s_{n+1} \vee s_n}$
s_{n+1}
s_{n-1}
s_0
z_{n-1}
z_{n-2}
z_0
(a)
(b)

Fig. 1. The architecture of the modulo $(2^n + 1)$ adders for diminished-1 encoded channel (a). Carry propagation increment stage (b).

Efficient parallel VLSI diminished-1 structures for modulo $(2^n + 1)$ two-operand addition have also been proposed recently [9].

Validity of the above diminished-1 addition of two $(n + 1)$ bit numbers, is demonstrated on following example.

Let $x' = 16$ and $y' = 10$. Then

$x' =$	10000
$y' =$	01010
$S_1 =$	011010

$S_2 =$	01010
$\overline{s_{n+1} \vee s_n} =$	0
$S' = x' + y' + 1$	01010

Thus, $S' = 10$, which can be verified to be true[1].

C. Diminished-1 modulo $(2^n + 1)$ multiplication

The diminished-1 multiplication is defined as [17], [14]:

$$Q' = \langle \langle x' \times y' \rangle_{2^n+1} + x' + y' \rangle_{2^n+1}. \tag{7}$$

The modulo $(2^n + 1)$ multiplication algorithm can be easily adapted for the diminished-1 number representation of input operands and output product. Thereby, the two additional terms x' and y' have to be added in the modulo carry save adder, resulting in small increasing of area and delay. The special case of $x'y' = 0$ has to be treated separately and the constant correction term be adapted.

The architecture of diminished-1 modulo $(2^n + 1)$ multiplication is presented in [17]. If the input residues are $(n + 1)$ bits wide, the partial products for modulo $2^n + 1$ multiple are arranged as n bits wide vectors. The partial product generation for inputs of 5 bits width is shown in Table II. Obviously, 4 bits $x_3x_2x_1x_0$ are required for the representation nonzero diminished-1 binary numbers. In this multiplication, the bits

TABLE II
PARTIAL PRODUCT GENERATION MOD $2^4 + 1$

2^6	2^5	2^4	2^3	2^2	2^1	2^0	
			$x'_0y'_3$	$x'_0y'_2$	$x'_0y'_1$	$x'_0y'_0$	$= pp_0$
		$x'_1y'_3$	$x'_1y'_2$	$x'_1y'_1$	$x'_1y'_0$	$\overline{x'_1y'_3}$	$= pp_1$
	$x'_2y'_3$	$x'_2y'_2$	$x'_2y'_1$	$x'_2y'_0$	$\overline{x'_2y'_3}$	$\overline{x'_2y'_2}$	$= pp_2$
$x'_3y'_3$	$x'_3y'_2$	$x'_3y'_1$	$x'_3y'_0$	$\overline{x'_3y'_3}$	$\overline{x'_3y'_2}$	$\overline{x'_3y'_1}$	$= pp_3$
			x'_3	x'_2	x'_1	x'_0	$= x'$
			y'_3	y'_2	y'_1	y'_0	$= y'$
			0	0	0	0	$= COR$

with weight greater than 2^3, which are to the left of the straight line, are complemented and repositioned to the right of the line.

The architecture of proposed modulo $2^n + 1$ multiplier for diminished-1 encoded channel is shown in Figure 2. Assuming the coefficient word length of 4-bits and input sample word length of 4-bits, in Fig. 2 is shown the hierarchical decomposition of Wallace tree logic. The partial sum are added by using five carry-save-adders (CSA) and modulo 17 adder which is realized as carry-propagate-adder with end-around-carry. Partial product is generated in parallel.

The principle of the proposed memoryless-based implementation of partial product generator is shown in Fig 3. It consists of n 2-to-1 multiplexers, where n is the input sample word length. The partial product is generated by connecting zero and coefficient value to the MUX data inputs, input data bits to the select input, and circular shifting of the MUX output $s - 1$ bits to the left, for $1 \leq s \leq n$. After circular shifting, $s - 1$ LSB bits are used as complement.

An implementation of the modulo 17 partial product generator for pp_2 is shown in Figure 4. The small circles above the register represent a complement of the input bit.

[1] $S' = \langle 16 + 10 \rangle_{17} + 1 = 10$

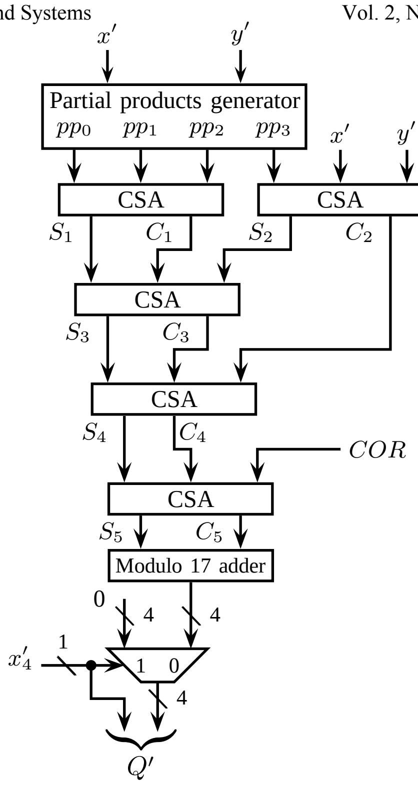

Fig. 2. Architecture of modulo $2^n + 1$ multiplier for diminished-1 encoded channel

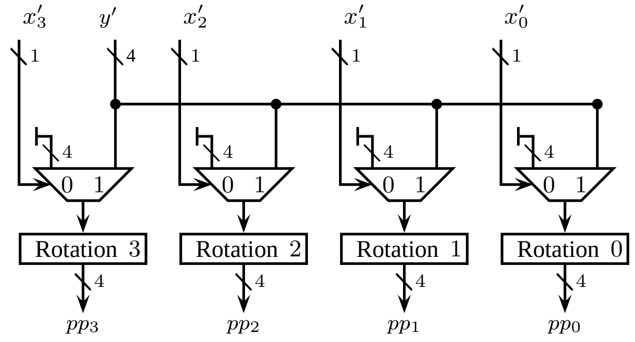

Fig. 3. Partial products generator.

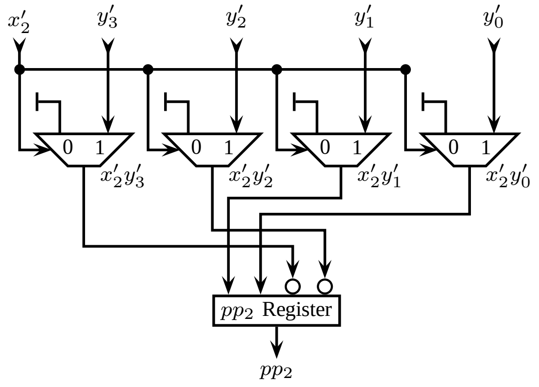

Fig. 4. Partial product pp_2 generator.

Validity of the above diminished-1 multiplication of two 4 bit numbers is demonstrated on following example. Let $x' = 9$ and $y' = 10$. Then

1010×1001

pp_0	=	1	0	1	0
pp_1	=	0	0	0	1
pp_2	=	0	0	1	1
pp_3	=	0	0	1	0
x'	=	1	0	1	0
y'	=	1	0	0	1
COR	=	0	0	0	0

The first carry-save-adder

pp_0	=		1	0	1	0
pp_1	=		0	0	0	1
pp_2	=		0	0	1	1
S_1	=		1	0	0	0
C_1	=	0	0	1	1	
						$\rightarrow$1

The second carry-save-adder

pp_3	=		0	0	1	0
x'	=		1	0	1	0
y'	=		1	0	0	1
S_2	=		0	0	0	1
C_2	=	1	0	1	0	
						$\rightarrow$0

The third carry-save-adder

S_1	=		1	0	0	0
C_1	=		0	1	1	1
S_2	=		0	0	0	1
S_3	=		1	1	1	0
C_3	=	0	0	0	1	
						$\rightarrow$1

The fourth carry-save-adder

S_3	=		1	1	1	0
C_3	=		0	0	1	1
C_2	=		0	1	0	0
S_4	=		1	0	0	1
C_4	=	0	1	1	0	
						$\rightarrow$1

The fifth carry-save-adder contains only half-adders since $COR = 0000$.

S_4	=		1	0	0	1
C_4	=		1	1	0	1
S_5	=		0	1	0	0
C_5	=	1	0	0	1	
						$\rightarrow$0

Finally

S_5	=		0	1	0	0
C_5	=		0	0	1	0
	=	0	0	1	1	0
						$\rightarrow$1
Q'	=		0	1	1	1

Thus $Q' = 7$, which can be verified to be true: $Q' = \Big\langle\langle 9 \times 10\rangle_{17} + 9 + 10\Big\rangle_{17} = 7$.

D. RNS to binary conversion

Consider the well-known 3-moduli set $\{m_1 = 2^n,\ m_2 = 2^n - 1,\ m_3 = 2^n + 1\}$ which has a dynamic range approximately equal to $3n$ bits. Wang, Jullien and Miller [18] show that the decoded binary number is obtained as

$$X = Y\,2^n + x_1, \tag{8}$$

where

$$Y = \Big\langle(-2^{2n-1} + 2^{n-1})x_3 + (2^{2n-1} + 2^{n-1})x_2 - 2^n x_1\Big\rangle_{2^{2n}-1} \tag{9}$$

This operation does not need any computation because equation (8) amounts to concatenation of Y obtained in (9) with x_1 as LSBs.

Residue number x_3 is encoded in diminished-1 as $x_3 = x_3' + 1$. Now let

$$A = \big\langle -2^n x_1\big\rangle_{2^{2n}-1}, \tag{10}$$

$$B = \big\langle(2^{2n-1} + 2^{n-1})x_2\big\rangle_{2^{2n}-1}, \tag{11}$$

$$C = \big\langle(-2^{2n-1} + 2^{n-1})(x_3' + 1)\big\rangle_{2^{2n}-1}. \tag{12}$$

To evaluate A, B and C, the following property is used: modulo $(2^s - 1)$ multiplication of a residue number by 2^t, where s and t are positive integers, is equivalent to t bit circular left shifting.

Assuming that x_1 is expressed in $2n$ bits, where n the most significant bits are zeros

$$x_1 = \underbrace{0\,0\ldots0}_{n}\underbrace{x_{1,n-1}x_{1,n-2}\ldots x_{1,0}}_{n} \tag{13}$$

and

$$A = \underbrace{\overline{x}_{1,n-1}\overline{x}_{1,n-2}\ldots\overline{x}_{1,0}}_{n}\underbrace{1\,1\ldots1}_{n} \tag{14}$$

where negative value of a number of modulo $(2^{2n} - 1)$ is the one's complement of that number.

Assuming that $2n$ bit expression of x_2 is given by:

$$x_2 = \underbrace{0\,0\ldots0}_{n}\underbrace{x_{2,n-1}x_{2,n-2}\ldots x_{2,0}}_{n} \tag{15}$$

it follows that

$$\begin{aligned} B &= x_{2,0}\underbrace{0\ldots0}_{n}\underbrace{x_{2,n-1}\ldots x_{2,1}}_{n-1} + 0\underbrace{x_{2,n-1}\ldots x_{2,0}}_{n}\underbrace{0\,0\ldots0}_{n-1} \\ &= x_{2,0}\underbrace{x_{2,n-1}\ldots x_{2,0}}_{n}\underbrace{x_{2,n-1}\ldots x_{2,1}}_{n-1}. \end{aligned} \tag{16}$$

The value of C, as given in (12), is:

$$C = \big\langle(-2^{2n-1} + 2^{n-1})x_3'\big\rangle_{2^{2n}-1} - K \tag{17}$$

where

$$K = 2^{2n-1} - 2^{n-1} = 0\underbrace{1\ldots1}_{n}\underbrace{0\ldots0}_{n-1}. \tag{18}$$

Let the $(n+1)$ bit expression of x_3' is given by:

$$x_3' = x_{3n}'\underbrace{x_{3,n-1}'x_{3,n-2}'\ldots x_{3,0}'}_{n} \tag{19}$$

or

$$x_3' = 2^n x_{3n}' + X_3'. \tag{20}$$

Therefore, parameter C given in (17) can be evaluated as follows: Substituting (20) into (17) and applying the above-mentioned property, it follows that

$$\begin{aligned}
&\langle(-2^{2n-1}+2^{n-1})X_3'\rangle_{2^{2n}-1}\\
=&\langle -(x'_{3,0}\underbrace{0\dots0}_{n}\underbrace{x'_{3,n-1}\dots x'_{3,1}}_{n-1})\\
&\quad+0\,\underbrace{x'_{3,n-1}\dots x'_{3,0}}_{n}\underbrace{0\dots0}_{n-1}\rangle_{2^{2n}-1}\\
=&\langle(\overline{x}'_{3,0}\,\underbrace{x'_{3,n-1}\dots x'_{3,0}}_{n}\underbrace{\overline{x}'_{3,n-1}\dots\overline{x}'_{3,1}}_{n-1})\\
&\quad+0\,\underbrace{1\dots1}_{n}\underbrace{0\dots0}_{n-1}\rangle_{2^{2n}-1}\\
=&\langle(\overline{x}'_{3,0}\,\underbrace{x'_{3,n-1}\dots x'_{3,0}}_{n}\underbrace{\overline{x}'_{3,n-1}\dots\overline{x}'_{3,1}}_{n-1})+K\rangle_{2^{2n}-1}
\end{aligned} \tag{21}$$

and

$$\langle(-2^{2n-1}+2^{n-1})2^n x'_{3n}\rangle_{2^{2n}-1}=0\,\underbrace{x'_{3n}\dots x'_{3n}}_{n}\underbrace{0\dots0}_{n-1}. \tag{22}$$

Adding (21), (22) and subtracting (18), it is obtained

$$\begin{aligned}
C=&x'_{3,0}\,\underbrace{x'_{3,n-1}\dots x'_{3,0}}_{n}\underbrace{\overline{x}'_{3,n-1}\dots\overline{x}'_{3,1}}_{n-1})+0\,\underbrace{x'_{3n}\dots x'_{3n}}_{n}\underbrace{0\dots0}_{n-1}\\
=&x'_{3,0}\,\underbrace{x_{or,n-1}\dots x_{or,0}}_{n}\underbrace{\overline{x}'_{3,n-1}\dots\overline{x}'_{3,1}}_{n-1},
\end{aligned} \tag{23}$$

where $x_{or,i}=x'_{3,i}\vee x'_{3n}$, for $0\le i\le n-1$ ($\vee$ denotes a logic OR operation). Note that $x_{3,i}$ and x_{3n}, can never be 1 at the same time.

To implement the modulo addition of three $2n$-bit numbers (A, B and C) efficiently, we may use $2n$ full-adders as carry-save-adders (CSA) to convert three $2n$ bit numbers into two. The carry-out from the most significant bit (c_{2n}) is fed to the least significant bit position (c_0). Then fast $2n$-bit carry-propagate-adder (CPA) with end-around-carry (EAC), is used to perform the modulo addition of two numbers to obtain the final result. The architecture is shown in Fig. 5.

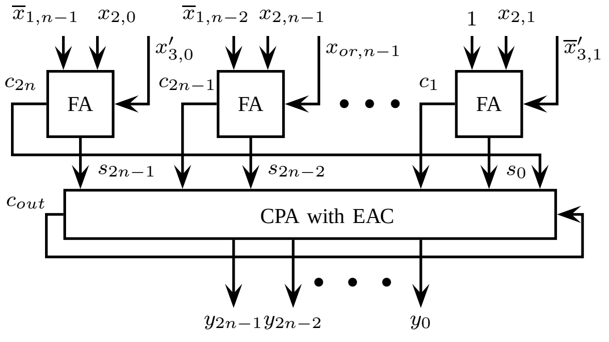

Fig. 5. The implementation of modulo $(2^{2n}-1)$ addition of three $2n$-bit numbers A, B and C.

E. An example

Consider the moduli set $\{16,15,17\}$ which is a special case of the moduli set $\{2^n,2^n-1,2^n+1\}$ for $n=4$. To convert the RNS number $X=(12,3,9)$ into its corresponding weighted binary representation, we have

$$\begin{array}{l|r}
x_1=12 & 1100\\
x_2=3 & 0011\\
x_3'=9-1=8 & 0\ 1000
\end{array}$$

Based on (14), (16) and (23), Y can be computed as

$$\begin{array}{rrl}
A= & 0011\ 1111 & \\
B= & 1001\ 1001 & \\
C= & 1100\ 0011 & +\\
\hline
\text{Sum vector }= & 0110\ 0101 & s_{2n-1}s_{2n-2}\dots s_0\\
\text{Carry vector }= & \underline{|1|}\ 0011\ 011 & c_{2n}c_{2n-1}\dots c_1\\
 & \longrightarrow 1 & \\
\hline
 & \underline{|0|}\ 1001\ 1100 & \\
c_{out}= & \longrightarrow 0 & +\\
\hline
\text{Correct Result: } Y= & 1001\ 1100 &
\end{array}$$

Finally, based on the equation (8), the final weighted binary number, X, can be simply calculated as

$$X=Y\ 2^4+x_1=1001\ 1100\ 1100=2508.$$

Thus, $X=2508$, which can be verified to be true.

III. Architecture of the Filter

Difference equations for each channel of FIR filter, implemented as RNS, can be defined as:

$$y_j(n)=\left\langle\sum_{i=0}^{N-1}\langle b_{i,j}x_j(n-i)\rangle_{m_j}\right\rangle_{mj},\ \text{for } j=1,2,3 \tag{24}$$

where $x_j(n)$ and $y_j(n)$ are the residue representations of the input and the output signals of the filter modulo m_j, respectively, and $b_{i,j}$, $i=0,1,2,\dots,N-1$ are the filter coefficients in RNS representation.

FIR filter can be implemented in a conventional scheme using delay elements. The delay elements actually pass the values delaying them by certain amount of time so that the signal values of the previous steps are multiplied with the corresponding coefficients. In this process, at each step, we need the computation of whole function.

The same FIR filter can be realized using Systolic Multiply-Accumulate architecture by implementing a pipelined Direct-Form filter [19], as depicted in Fig. 6. In this technique, the computation is partitioned into smaller parcels that can be assigned to a series of different concurrent processing elements in such a way as to achieve advantage in speed.

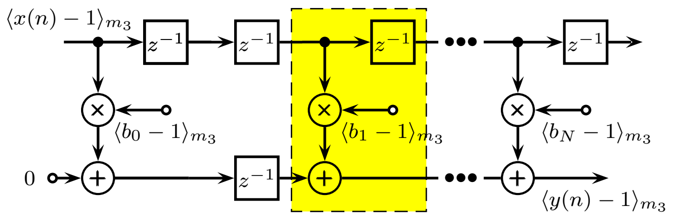
$\langle x(n)-1\rangle_{m_3}$
z^{-1}
z^{-1}
z^{-1}
z^{-1}
$\langle b_0-1\rangle_{m_3}$
$\langle b_1-1\rangle_{m_3}$
$\langle b_N-1\rangle_{m_3}$
0
z^{-1}
$\langle y(n)-1\rangle_{m_3}$

Fig. 6. Systolic realization of diminished-1 encoded channel ($m_3=2^n+1$) of Nth-order nonrecursive digital filter. One processing element is dashed part of figure.

This linear phase lowpass filter, designed by means of published program based on the Parks-McClellan algorithm, is normalized so that the passband edge is 0.4π, stopband edge is

0.5π and the minimum stopband attenuation is 32.625dB. The filter coefficients are shown in Table III for double precision (the IEEE 754 standard) and for 10-bit precision in integer notation.

TABLE III
THE 31TH-ORDER FIR LOWPASS FILTER COEFFICIENTS FOR MODULI SET $(2^n-1, 2^n 2^n+1)$, WITH $n=6$.

Coefficients b_i	Double precision	Int.	RNS number [a]		
			$b_{i,1}$	$b_{i,2}$	$b'_{i,3}$
$b_0 = b_{24}$	–0.0005790732070	0	/	/	/
$b_1 = b_{23}$	0.0107143423917	5	5	5	4
$b_2 = b_{22}$	0.0056224531799	3	3	3	2
$b_3 = b_{21}$	–0.0121381434840	–6	58	57	58
$b_4 = b_{20}$	–0.0186695715150	–10	54	53	54
$b_5 = b_{19}$	0.0085434429402	4	4	4	3
$b_6 = b_{18}$	0.0385495109566	20	20	20	19
$b_7 = b_{17}$	0.0119972274970	6	6	6	5
$b_8 = b_{16}$	–0.0606997415311	–31	33	32	33
$b_9 = b_{15}$	–0.0684021145565	–35	29	28	29
$b_{10} = b_{14}$	0.0782461959053	40	40	40	39
$b_{11} = b_{13}$	0.3044489251280	156	28	30	25
b_{12}	0.4150604524227	213	21	24	17

[a]The RNS number $b'_{i,3} = \langle b_i \rangle_{m_3}$ is diminshed-1 coded.

Since integer values of coefficients b_0 and b_{31} are equal to zero, the filter length is reduced to $N = 30$, as it is shown in Table III.

Integer values in the third column in Table III are transformed from floating point value (second column) in two steps. The first step is the conversion of floating point filter coefficients, b, into binary string `b` using two MATLAB© functions, `Q_1=quantizer('round',Format)` and `b_binary=num2bin(Q_1,b)`. Value of parameter `Format` creates parameters of binary numbers: `[wordlength,fractionlength]` for signed fixed-point mode. For 10-bit precision format these parameters are wordlength=10 and fractionlength=9.

This paper investigates binary-to-residue converter for the modulo set $\{63, 64, 65\}$ with diminished-1 encoded $2^n + 1$ channel. In the following example we describe the fixed point-to-residue number system conversion of coefficient b_1. Double precision of filter coefficient b_1 is -0.012138143484039 which is converted into binary number `b_binary=1111111010`, then into integer number `b_int=-6`, and finally into RNS number `b_RNS=(58,57,59)`.

The filter generated by RNS moduli set $\{2^n-1, 2^n, 2^n+1\}$ with diminished-1 encoded channel, for $n = 6$ provides the dynamic range $M = 262\ 080$. Hereby the filter can operate in this range and provide better bit efficiency than existing standard RNS based filters.

A. Filter Performance

The simulation, performed in Matlab (R2010a), depicts the effects of this approach on the filter design.

Assume that the data sequence is quantized to 8 bits (including sign) and that filter must be implemented without

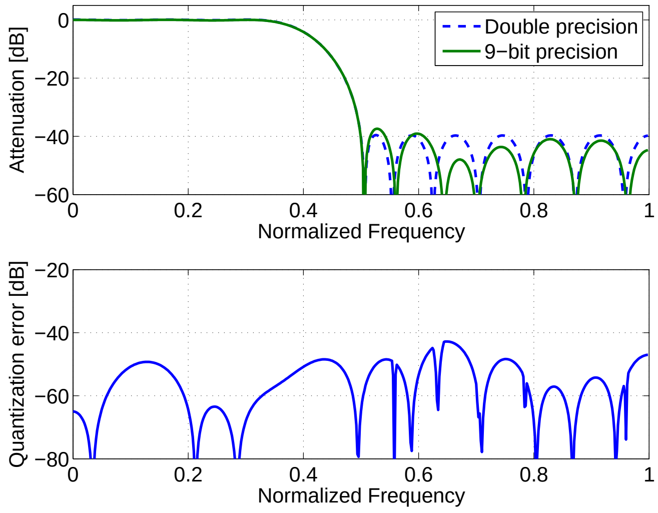

Fig. 7. Attenuation response (top) and quantization error for the coefficient rounding to 9-bit (bottom).

rounding error. An absolute upper bound of filter response $|y_{\mathrm{m}}(n)|$ is given by equation (25)

$$|y_{\mathrm{m}}(n)| \le \max\{|x(n)|\} \sum_{k=1}^{23} |b_k| = 216320 \approx 17.72 \text{ bits}. \quad (25)$$

The moduli set $\{63, 64, 65\}$ provides a dynamic range of 17.99 bits, which is adequate for the most practical cases since the bound of 17.87 bits, given by (25), is extremely pessimistic [20].

Figure 8 shows impulse response of the RNS channels. In $\{64, 63, 65\}$ proposed residue number system unit sample is

$$\delta(n) = \begin{cases} (51, 54, 47), & \text{for } n = 0 \\ (0, 0, 64), & \text{for } n > 0. \end{cases} \quad (26)$$

First two channels are normally encoded, but the third is diminished-1 encoded.

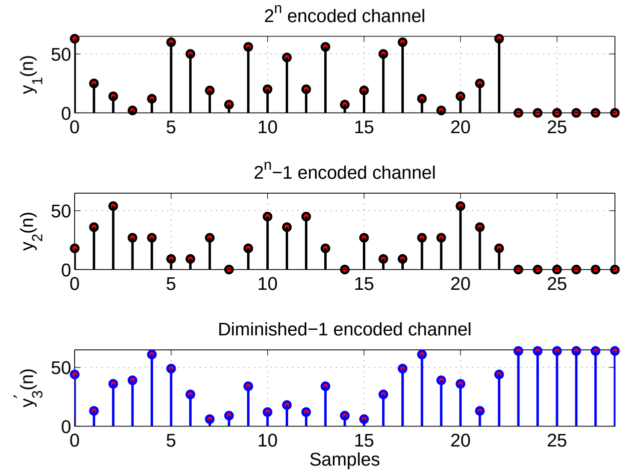

Fig. 8. The impulse response of the RNS channels: 2^n channel, above; $2^n - 1$ in the middle; $2^n + 1$ diminished-1 encoded channel, down.

Response of each channel is symmetric about 11-th sample. In diminished-1 encoded channel zero is coded with 64.

The impulse response of this digital filter is shown in Figure 9, where samples are given in integer form. The required number of bits is the sum of the coefficient bits and data bits. In this design it is 17 bits. Thus, dividing by 2^{17} the integer response is transformed into fixed point response.

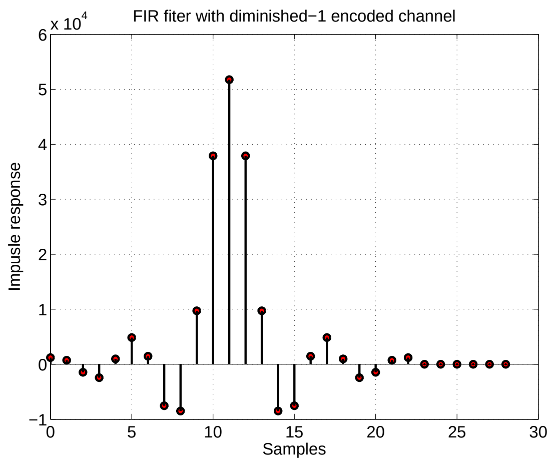

Fig. 9. Impulse response of the RNS lowpass filter.

IV. Conclusion

The design of a finite impulse response digital filter in a residue number system has been presented. The RNS coding technique with diminished-1 encoded channel is attractive for FIR filters which require only multiplication and addition because these operations are very fast in an RNS. The architecture of all building blocks, except residue-to-binary converter with diminished-1 encoded channel and architectures for modulo multiplication, has already been discussed in previous papers. The architecture for reverse convertor that includes diminished-1 encoded channel, which uses only binary adders without memory blocks, is proposed in this paper. To achieve high speed, new partial product generator combining with the Wallace tree is adopted for the multipliers.

Future research includes the extension of this study to Xilinx chips, the power-figure measurement and a full characterization of each design option at layout level.

Acknowledgment

This work was supported by the Serbian Ministry of Education and Science in the frame of the project TR 33008 and III 44004.

References

[1] W. K. Jenkins and B. Leon, "The use of residue number systems in the design of finite impulse response digital filters," *IEEE Trans. on Circuits and Systems*, vol. CAS-24, no. 4, pp. 191–201, Apr. 1977.

[2] M. A. Soderstrand and B. Sinha, "A pipelined recursive residue number system digital filter," *IEEE Trans. on Circuits and Systems*, vol. CAS-31, no. 4, pp. 415–417, Apr. 1984.

[3] P. A. Regalia, S. K. Mitra, P. Vaidyanathan, M. K. Renfors, and Y. Neuvo, "Three-structured complementary filter bank using all-pass sections," *IEEE Trans. on Circuits and System*, vol. CAS-34, no. 12, pp. 1470–484, Dec. 1987.

[4] T. K. Shahana, R. James, B. Jose, K. Jacob, and S. Sasi, "Performance analysis of FIR digital filter design: RNS versus traditional," in *International Symposium on Communications and Information Technologies. ISCIT'07*, Kochi, Kerala, India, Oct. 17–19, 2007, pp. 1–5.

[5] R. Conway and J. Nelson, "Improved RNS FIR filter architectures," *IEEE Trans. on Circuits and Systems-II: Express Briefs*, vol. 51, no. 1, pp. 26–28, Jan. 2004.

[6] L. M. Leibowitz, "A simplified binary arithmetic for the Fermat number transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-24, no. 5, pp. 356–359, Oct. 1976.

[7] R. Zimmermann, "Efficient VLSI implementation of modulo $(2^n \pm 1)$ addition and multiplication," in *Proceedings of the 14th IEEE Symposium on Computer Arithmetic*, Adelaide, Australia, Apr. 1999, p. 158167.

[8] Z. Wang, G. A. Jullien, and W. C. Miller, "An efficient tree architecture for modulo 2^n+1 multiplication," *VLSI Signal Processing*, vol. 14, no. 3, pp. 241–243, Mar. 1996.

[9] H. T. Vergos, C. Efstathiou, and D. Nikolos, "High speed parallel-prefix modulo $2^n + 1$ adders for diminished-one operands," in *Proceedings of 15th IEEE Symposium on Computer Arithmetic*, Vail, CO, USA, June 11–13, 2001, pp. 211–217.

[10] H. Vergos and C. Efstathiou, "A unifying approach for weighted and diminished-1 modulo $2^n + 1$ addition," *IEEE Trans. on Circuits and Systems-II: Express Briefs*, vol. 55, no. 10, pp. 1041–1045, Oct. 2008.

[11] R. Chaves and L. Sousa, $\{2^n+1, 2^{n+k}, 2^n-1\}$*: a new RNS moduli set extension*. IEEE, 2004, pp. 210–217. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1333279

[12] C. Efstathiou, H. T. Vergos, and D. Nikolos, "Handling zero in diminished-one modulo 2^n+1 adders," *Int. J. Electronics*, vol. 90, no. 2, pp. 133–144, Feb. 2003.

[13] C. Efstathiou, I. Voyiatzis, and N. Sklavos, "On the modulo $2^n + 1$ multiplication for diminished-1 operands," in *Proc. of the 2nd Int. Conf. on Signals Circuits and Systems (SCS 2008)*.

[14] J. W. Chen, R. H. Yao, and W. J. Wu, "Efficient modulo $2^n + 1$ multipliers," *IEEE Transactions on Circuits and Systems*, vol. 19, no. 12, pp. 2149–2157, Dec. 2011.

[15] A. Hämäläinen, M. Tommiska, and J. Skyttä, "6.78 gigabits per second implementation of the IDEA cryptographic algorithm," in *Proceedings of the 12th Conference on Field-Programmable Logic and Applications (FPL 2002)*, M. Glesner, P. Zipf, and M. Renovell, Eds. Montpellier, France: Springer-Verlag, Sept. 2–4, 2002, pp. 760–759.

[16] J.-L. Beuchat, "Some modular adders and multipliers for field programmable gate arrays," in *Proceedings of 17th International Symposium on Parallel and Distributed Processing, IPDPS'03*. Los Alamitos, CA, USA: IEEE Computer Society, Apr. 22–26, 2003, pp. 190–197.

[17] C. Efstathiou, H. T. Vergos, G. Dimitrakopoulos, and D. Nikolos, "Efficient diminished-1 modulo $2^n + 1$ multipliers," *IEEE Transactions on Computers*, vol. 54, no. 4, pp. 491–496, Apr. 2005.

[18] Z. Wang, G. Jullien, and W. Miller, "An improved residue-to-binary converter," *IEEE Transactions on Circuits and Systems I*, vol. 47, no. 9, pp. 1473–1440, Sept. 2000.

[19] A. Antoniou, *Digital Signal Processing: Signals, Systems, and Filters*. New York: McGraw-Hill, 2006.

[20] W. K. Jenkins and B. Leon, "The use of residue number systems in the design of finite impulse response digital filters," *IEEE Trans. on Circuits and Systems*, vol. CAS-24, no. 4, pp. 191–201, Apr. 1977.

Dragana Živaljević was born in Niš, Serbia, Yugoslavia, on February 14, 1972. She enrolled in the Faculty of Electronic Engineering, University of Niš, Serbia and received B.Sc. degree in 2001. Since 2002 she has been working as a lecturer and research assistant at the Department of Theoretical Electrical Engineering at the Faculty of Electronic Engineering in Nis. She is involved in conducting auditing exercises for courses Electrical Engineering I and II, Electrical Circuits Theory, Electromagnetic, Digital signal processing and Signals and systems.

Negovan Stamenković was born on march 30, 1979. He received the B.Sc. degree and M.Sc. degrees in electronics and telecommunications from the Electrical and Engineering Department at Faculty of Technique in Kosovska Mitrovica. He received Ph.D. degree in electronics and telecommunications from Faculty of Electronic Engineering at University of Niš. He was elected professor the Faculty of Sciences and Mathematics in Kosovska Mitrovica. His research is based on signal processing residue numerical system.