Hardware Implementation of Fir Filter Based on Number-theoretic Fast Fourier Transform in Residue Number System

V.M. Amerbaev, R.A. Soloviev and D.V. Telpukhov*

Institute for Design Problems in Microelectronics of RAS, Zelenograd, Moscow

Abstract: New approach to hardware FIR filter implementation is described in this paper. Method includes the conversion to the frequency domain, as well as the principles of residue number systems and number-theoretic transforms. Parameterized RTL IP-core generators were implemented for both conventional and developed methods. These IP generators were used to implement different devices for different filter orders and input widths. Filters were synthesized, and resulting time and hardware evaluation allow one to consider the effectiveness of the method compared with the conventional realization of FIR filter.

Keywords: Convolution theorem, finite impulse response filter, number theoretic fast fourier transform, residue number system, verilog.

INTRODUCTION

Filtering with finite impulse response (FIR) is a basic procedure in various digital signal processing devices involved in such research areas as speech processing, radar signal processing, filtering all sorts of noise in a wide range of human activities.

To improve performance of program filters, designers often move to frequency domain and use Fast Fourier Transform procedures. This method is based on the so called convolution theorem, theorem, which allows reducing the number of operations for higher-order filters. Nevertheless, the program implementation of FIR filters in signal processors or general purpose processors is strictly sequential, that cannot provide high performance in many practical cases. Often the bandwidth requirements in real devices can be met only for hardware implementation of constant-coefficient filters. Sacrificing flexibility in choosing coefficients as well as hardware costs, it is possible to provide the required level of performance using pipelined architecture for FIR filters implementation. The problem of constructing hardware FIR filters with fixed coefficients has been actively discussed in foreign publications in recent years [1, 2].

The process of further increase of the FIR filter's performance involves the use of mathematical apparatus of residue number system (RNS) arithmetic which has repeatedly proved its effectiveness in the field of digital signal processing, in particular for digital filters design [3].

Besides using RNS principles, we propose to combine the idea of computation in frequency domain with the principles of parallel data processing in the same hardware FIR filter. In addition, RNS implementation of FIR filter using number-theoretic Fourier transform is proposed, which provides increased performance and better accuracy.

*Address correspondence to this author at the Institute for Design Problems in Microelectronics of RAS, Zelenograd, Moscow; Tel: 8(499)7299890; Fax: 8(499)7299208; E-mail: nofrost@inbox.ru

PRELIMINARIES

Conventional Implementations of FIR Filters

Filter with finite impulse response, in essence, is no more than a linear convolution of the input sequence of some digital points with the sequence of filter coefficients. The task of choosing particular set of filter coefficients can be effectively solved by modern software tools and is not considered in our work. Abstracting from the choice of the coefficient values, we turn directly to the computation of linear convolution. The formula for its calculation is as follows:

$$s(n) = a * b = \sum_{m=0}^{n} a(m) \cdot b(n-m), n = 0 \ldots N + M - 2 \qquad (1)$$

where $a(n)$, $n = 0 \ldots N-1$, and $b(n)$, $n = 0 \ldots M-1$ are discrete signals; $s(n)$ is the linear convolution of these signals.

To compute the linear convolution, the signals $a(n)$ and $b(n)$ are shifted relative to each other, then point-wise multiplied and folded. It is assumed that $a(n) = 0$ at $n < 0$ and $n > N$, and $b(n) = 0$ at $n < 0$ and $n > M$.

Architectures for computing linear convolution can be different. There are several types of architectures:

- Sequential
- Parallel
- Sequential-parallel

Sequential schematics is characterized by a small number of computing units, intensive memory exchange and low performance. In the extreme, this circuit consists of a multiplier-accumulator and a control device which provides the

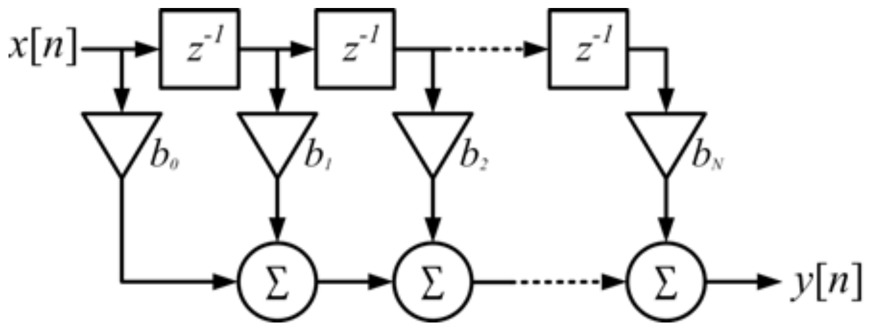

Fig. (1). Canonical form of FIR filter.

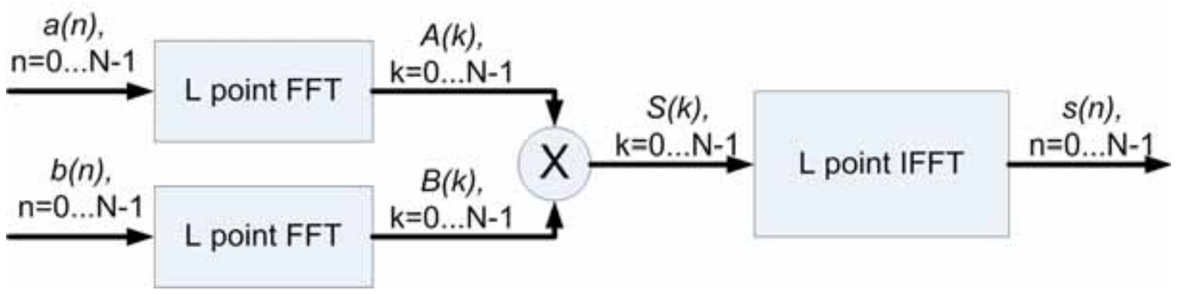

Fig. (2). Block diagram of the cyclic convolution calculation in frequency domain.

necessary loading of coefficients from memory. In this case, N clock cycles are required to calculate one output count. This method is implemented as software in signal processors or general purpose computers.

If the performance of digital signal processor (DSP) is not enough, the filter is implemented as hardware using parallel architectures. Parallel schematics exploit pipelining method, separating the pipeline stages by registers. Canonical form of FIR filter looks as follows (Fig. **1**):

The advantage of this architecture is its speed and ability to work in real time. Its disadvantage is the considerable increase of hardware costs.

Implementation of FIR Filters in Frequency Domain

Besides the time-domain implementations, frequency domain implementation is also possible. It is based on the so-called convolution theorem, the essence of which is as follows: the spectrum of the cyclic convolution equals the product of the spectra of the convolved signals (convolution in one domain equals point-wise multiplication in the other domain): $S(k) = A(k) * B(k)$, where $A(k)$ and $B(k)$ are the spectra of the convolved signals, and $S(k)$ is the spectrum of the cyclic convolution of two signals (Fig. **2**).

Thus, instead of directly implementing the convolution according to the formula, we can convert the signals into frequency domain using fast Fourier transform (FFT), multiply them point-wise, and then do reverse conversion. In this case the result corresponds to the cyclic convolution [4]. Moreover, if we add zeros to the sequences, cyclic convolution calculation becomes equivalent to calculation of the required linear convolution. The advantage of this method is the reduction of operations number, because using FFT schematics we need only $n*\log_2(n)$ operations, which gives a gain for large values of *n*.

However, there are some drawbacks. In most cases, the input signal is interpreted as a signal of infinite duration, which is filtered using a limited set of coefficients. Parallel circuit successfully copes with this task in "online" mode, but as regards the methods based on FFT, it is not so easy. Even if we create a converter for conditional infinite number of points, the circuit latency also goes to infinity, because for FFT, to start giving out the resulting values, all needed values should be uploaded beforehand. To solve this problem, various "overlap-add" methods have been developed. They allow to process small size segments and then combine them to get the necessary convolution for the theoretically infinite sequence of points at the output [5].

FIR Filters Implementation *via* Residue Number System

For further improvement of performance, reliability, accuracy and power consumption, methods based on RNS arithmetic are often used. Using the Chinese remainder theorem, it is possible to decompose the filter structure into a number of parallel independent residue channels, each of which is a FIR filter over a finite field. Such parallelization reduces number of digits of the operands, that improves operating speed and power consumption. Also, adding some redundancy, RNS arithmetic allows increasing device reliability.

Structure of RNS calculators is determined by the set of moduli $\{m_1, m_2, \dots, m_N\}$, that are pairwise coprime. Dynamic range of the RNS is characterized by an interval [0,M), where $M = m_1 \cdot m_2 \cdot \dots \cdot m_N$. Chinese remainder theorem guarantees unique representation of integer numbers from the dynamic range as the set of remainders of the number divided by the moduli from the given set, and also defines the method of restoring the number from its remainders. Thus, the number has the following RNS notation:

$(|X|_{m1}, |X|_{m2}, ..., |X|_{mN})$

where $|X|_{m_i}$ is the remainder of the division of X by the modulus m_i.

Arithmetic operations are performed componentwise Z=X°Y, where X=($x_1, x_2, ..., x_r$), Y=($y_1, y_2, ..., y_r$), Z=($z_1, z_2, ..., z_r$), and $z_i = x_i \circ y_i$ for i = 1,…,r. Symbol ° denotes any of the RNS operations: addition, subtraction, or multiplication for corresponding modulus. Thus, the calculations are performed in parallel, independently for each residue channel, that eventually leads to the process acceleration.

One of the main difficulties regarding the practical implementation of RNS-based devices is the problem of effective implementation of forward and reverse converters - devices that convert information from binary to RNS notation and backwards. Building of reverse converters is especially hard because it requires simultaneous addressing all components of RNS notation. For implementing of FIR filters, in this paper we use efficient architectures of forward and reverse converters that were proposed in the previous works [6, 7].

DEVELOPMENT OF RNS-BASED FIR FILTER USING THE CONVOLUTION THEOREM OVER FINITE FIELDS

As it was already stated above, there are many ways of hardware implementation of FIR filters: sequential, parallel, in frequency/time domain, etc. RNS arithmetic provides means for structure parallelization at the level of arithmetic operations, and it can be used for the design of a FIR filter implemented by any method. However, the most promising is the use of RNS-based parallel implementation in frequency domain. There are several prerequisites for this conclusion, the main of which is that for finite fields there is an analogue of the convolution theorem. More precisely, there exists a transform over finite fields similar to Fourier transform, for which convolution theorem over finite fields holds and all symmetric properties are met, that allows using any fast algorithm for computing this transform. The conversion formula is as follows:

$$A_k = \left| \sum_{n=0}^{2^t-1} \left| a_n * \gamma^{nk} \right|_p \right|_p$$

Here A_k - is the number-theoretic spectrum of the signal, $a_n$is the signal itself, p is the characteristic of the finite field, γ - is the primitive root with index 2^t [8]. If the number p is of the form $p = q \cdot 2^t + 1$, then the primitive root with index 2^t always exists and equals $\gamma = \omega^q$.

For a prime modulus *m***,** there is at least one primitive root $\omega \le p - 1$, such that the set $|\omega^i|_m : i = 0,1,2,...,m-2$ forms all nonzero residues modulo *m*. The following example shows that $\omega = 3$ is a primitive root modulo $m = 7$:

$|3^0|_7 = 1$ $\quad |3^3|_7 = 6$

$|3^1|_7 = 3$ $\quad |3^4|_7 = 4$

$|3^2|_7 = 2$ $\quad |3^5|_7 = 5$

Exponents of the primitive root by an arbitrary modulo are called "indexes".

Thus, the said conversion is an analog of the discrete Fourier transform for trigonometric basis, where the twiddle factors $e^{ij\frac{2\pi}{N}}$ are replaced by γ^{ij}, and the operations of addition and multiplication of complex numbers – by the corresponding operations over *GF(p)*.

Besides the obvious advantages such as of computations paralleling, we achieve better accuracy, since the calculations are carried out in finite fields with integer twiddle factors that eliminates representation errors typical for standard trigonometric method. [4]

The RNS-based FIR filter implementation with the use of number-theoretic fast Fourier transform is presented in the (Fig. **3**).

The basic structure of the filter consists of forward and reverse converter units, as well as N residue channels of the same type that differ only in characteristic of the finite field over which all arithmetic calculations are carried out. All moduli must have the same binary rank, i.e. they must all be of the form $p_i = q_i \cdot 2^t + 1$. This fact guarantees that a primitive root γ_i with index 2^t exists in every finite field, that allows to form RNS-based FFT of length 2^t. This parameter is the basis of FIR filter as a whole. It also determines the length of input data segments of the infinite sequence to be processed in residue channels and then "glued" in OVERLAP ADDITION block. Moreover, this parameter affects the filter order, and if it equals $p_i = q_i \cdot 2^t + 1$, the number of filter coefficients must equal 2^{t-1}.

The modular channel consists of a zero padding block, forward FFT block, a unit for multiplication by filter coefficients, inverse FFT block (INV_FFT) and a module for data segments combination. FIR filter behavior in frequency domain implies that the relevant number of zeros are added to input data segments for correct “gluing” of the results in the block for combination of the processed data segments. Taking into consideration the pipelined device architecture, it is necessary to reduce data input frequency in half. In other words, internal FIR frequency F_{clk} should be two times higher than discretization frequency F_s. For this purpose frequency divider block is added to the schematic (Fig. **4**).

Overlap addition block combines parts of processed data to make an infinite output sequence of filtered result. Inverse Fourier transform produces blocks of data of length 2^t, each of which must be divided into two parts. These two sequences must be combined through pairwise summation. Let’s consider t = 3, and show the overlapping process on (Fig. **5**).

Now it is clear that for 2^t samples of input data overlap addition block produces 2^{t-1} output samples. It means that we must use discretization frequency F_s for output data. Verilog code for overlap addition block is given below.

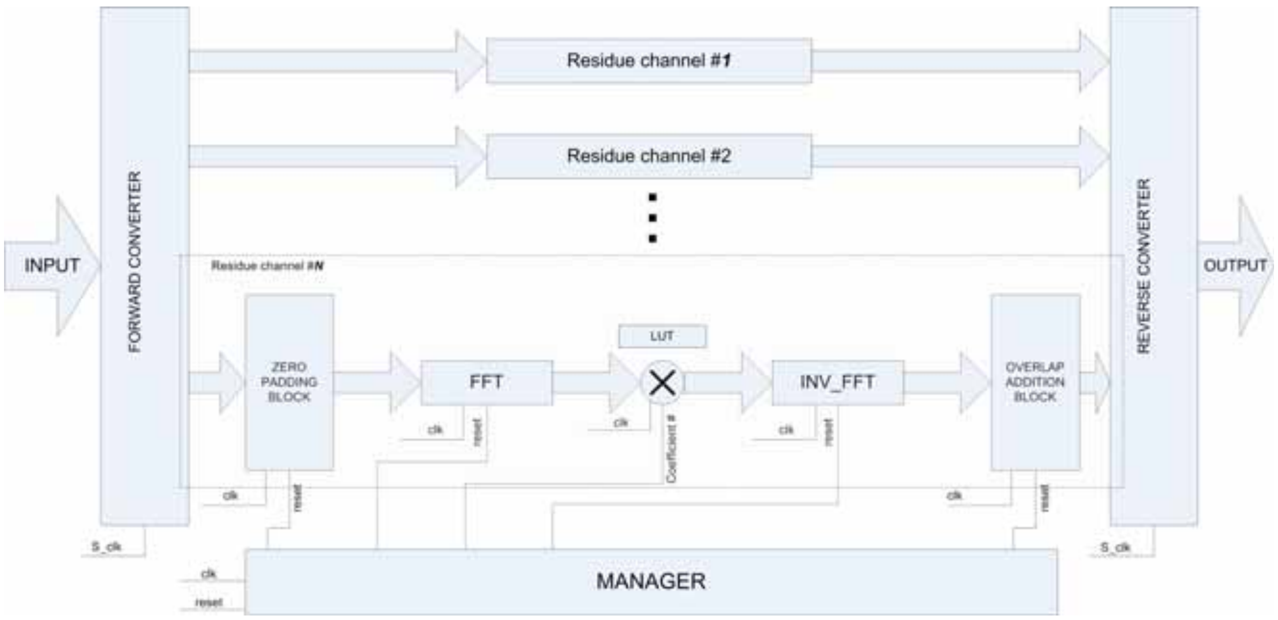

Fig. (3). Block schematic diagram of FIR filter in frequency domain.

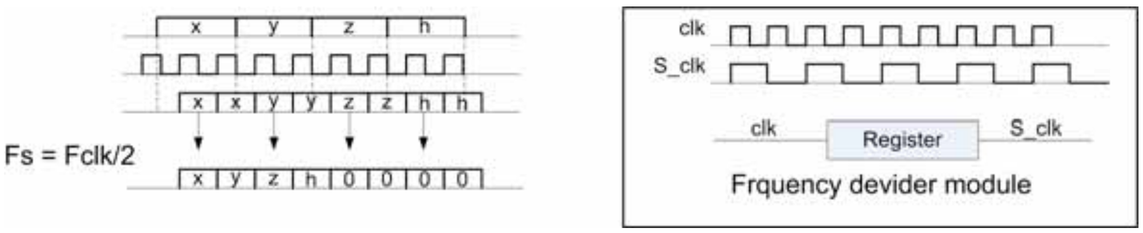

Fig. (4). Zero padding block and frequency divider block.

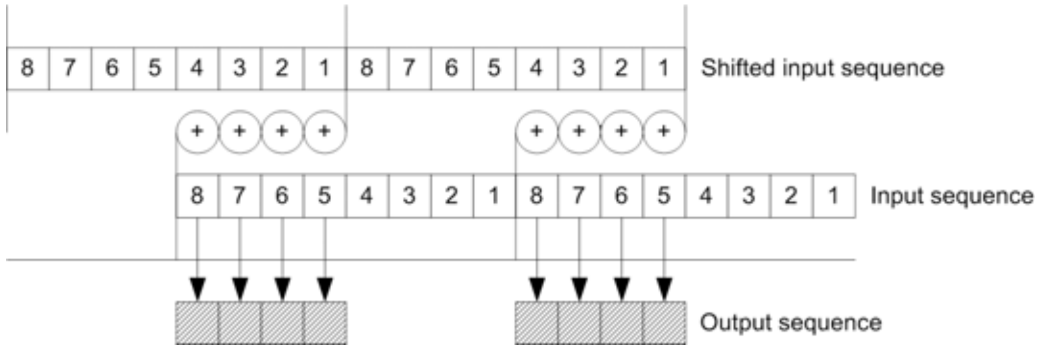

Fig. (5). Synthesis results for binary filter in the canonical form.

```
module overlap_add_41 (dataout, datain, clk, reset,
clk_enable);
        parameter WIDTH=6;
        input  [WIDTH-1:0] datain;
        input  clk, reset, clk_enable;
        output [WIDTH-1:0] dataout;
        reg clk_2;
        wire [WIDTH-1:0] shift1_out, adder_out;
```

```
            // Block Statements
        always @ (posedge clk or posedge reset)
        begin
            if (reset)
                clk_2 <= 0;
            else
                clk_2 <= ~clk_2;
        end
```

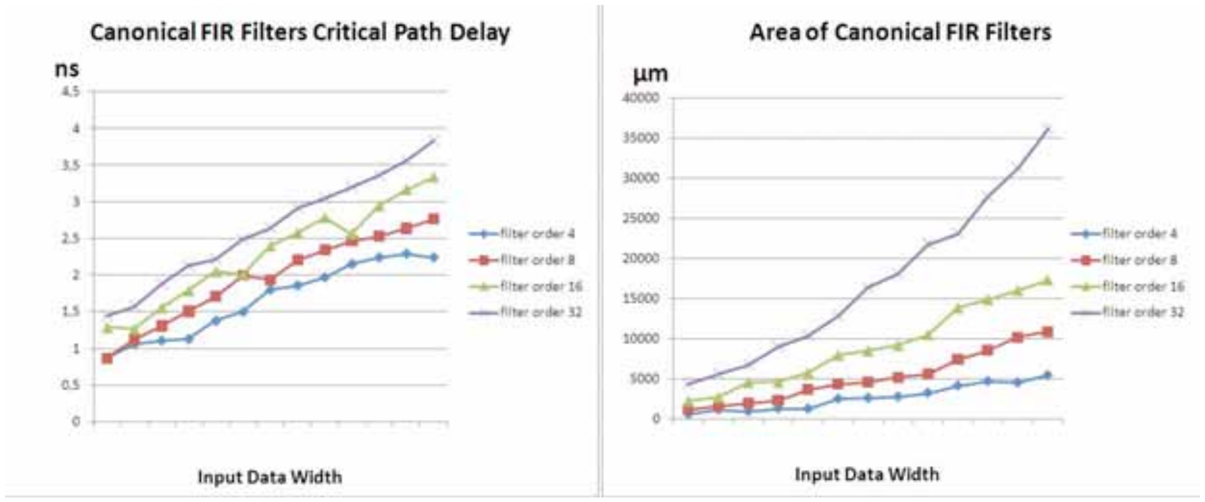

Fig. (6). Synthesis results for RNS filter based on convolution theorem over finite field.

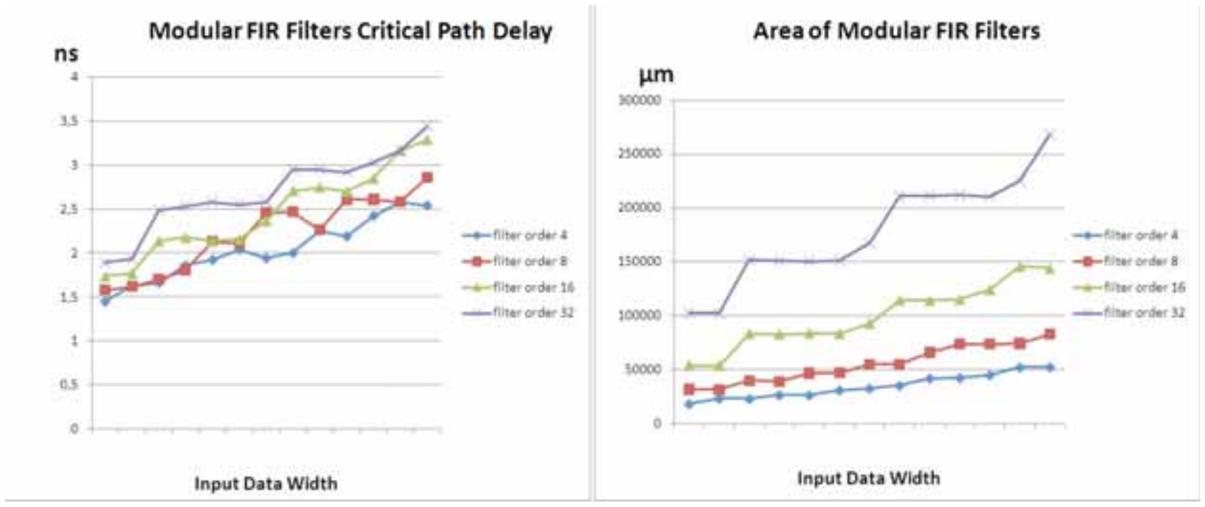

Fig. (7). Comparison of RNS and binary schematics for fixed filter orders.

```
        shift_reg_4 #(WIDTH) shift1 (shift1_out, datain,
clk, reset, clk_enable);

        mod_adder #(41, 6) add (adder_out, shift1_out, da-
tain);

        slow_down_4 #(WIDTH) slow (dataout, ad-
der_out, clk, clk_2, reset, clk_enable);

    endmodule
```

All internal blocks including Fourier transform block (FFT) are maximally pipelined to achieve the smallest clock frequency. Control unit MANAGER provides for the synchronization of device parts and device as a whole. It is also responsible for supplying the address of the cell holding the filter coefficient to the memory connected to the multiplier.

SYNTHESIS RESULTS

The result of the design is the IP core implementing RTL circuit description in Verilog HDL. All IP cores, developed during related work are available on the author's web site [9]. IP core parameters are the filter coefficients and the input data digit capacity. To evaluate the efficiency of the proposed approach, experiment was carried out on the synthesis of RNS-based FIR filters using Synopsys CAD tools and Nangate 45 nm Open Cell Library. The order of the filter was chosen in the range of 4 to 32 points. Number of bits is in the range of 4 - 16. Critical path delay and circuit area were evaluated. Also, for the same set of parameters FIR filter in canonical form was designed. Results are presented in Figs. (**6**, **7**) and Table **1**.

CONCLUSION

The synthesis results show that the proposed schematic outperforms the traditional variant with regard to clock frequency for large-scale input data and higher filter order, however, it requires sufficiently greater area. And if at first glance, the results may seem disappointing – actually it is not so. RNS-based filters have some advantages which are not reflected in the presented diagrams. First, the filtered output signal of the RNS block based on the FFT over a fi-

Table 1. Comparison of RNS and binary schematics for fixed filter orders.

Input Data Width	Timing Comparison, ns		Area Comparison, µm	
	Canonical FIR Filter	RNS FIR Filter	Canonical FIR Filter	RNS FIR Filter
4	1.45	1.89	4326	102768
5	1.58	1.93	5607	102456
6	1.88	2.49	6667	151908
7	2.13	2.53	8996	151088
8	2.22	2.58	10374	150621
9	2.5	2.55	12894	151169
10	2.64	2.58	16386	167788
11	2.92	2.95	18050	211303
12	3.06	2.95	21777	211460
13	3.21	2.92	23062	212172
14	3.37	3.03	27726	210621
15	3.57	3.16	31257	225272
16	3.84	3.44	36158	268385

nite field is much more accurate than the output signal of the conventional binary filter. This is due to the fact that for FFT over a finite field twiddle coefficients are the elements of this finite field, *i.e.* integer numbers. Thus, we avoid errors in the representation of the coefficients. Another advantage of the proposed approach is the fact that by adding extra modular channels we are able to control and even correct errors that arise during the computation. Future developments will be directed to the implementation of sequential RNS filters based on FFT over a finite field.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

Declared none.

REFERENCES

[1] P. Patronik, K. Berezowski, S. Piestrak, J. Biernat, and A. Shrivastava. “Fast and energy-efficient constant-coefficient FIR filters using residue number system,” in 17th *IEEE/ACM International Symposium on Low Power Electronics and Design*(*ISLPED'11*), 2011.

[2] A. Shrivastava, K. S. Berezowski, S. J. Piestrak, and R. Chokshi. “Exploiting residue number system for power-efficient digital signal processing in embedded processors,” in *IEEE/ACM International Conference on Compilers, Architecture, and Synthesis for embedded Systems (CASES)*, France, 2009.

[3] M.A. Soderstrand, W. K. Jenkins, G. A. Jullien, and F. J. Taylor, Eds., *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*. Piscataway, NJ: IEEE Press, 1986.

[4] V. Amerbaev, D. Telpukhov, R. Solovyev, and A. Stempkovskiy. “Efficient Calculation of Cyclic Convolution by Means of Fast Fourier Transform in a Finite Field,” Proc. of 11th *East-West Design & Test Symposium*, 2013.

[5] L. R. Rabiner, and B. Gold, *Theory and application of digital signal processing.* Englewood Cliffs, NJ: Prentice-Hall. pp. 63-67, 1975.

[6] R. A. Solovyev, and D. V. Telpukhov. “Hardware implementation of calculation of remainder from integer division for large-scale input data in residue arithmetic,” *Izvestiya VUZ Journal. Electronics*. no. 4, pp. 75-83, 2013. (In Russian)

[7] D.V. Telpukhov. “Building of reverse converters for digital signal processing devices basing on RNS logarithmetic,” *Informational Technologies Journal*, no.4. 2011. (In Russian)

[8] M. Vinogradov, *Elements of Number Theory*. Dover publications Inc., 1954.

[9] www.vscripts.ru/verilog_generator_modular_fir_filter.php

Received: March 20, 2014 Revised: June 25, 2014 Accepted: July 02, 2014