POZNAN UNIVERSITY OF TECHNOLOGY ACADEMIC JOURNALS
No 76 Electrical Engineering 2013

Maciej CZYŻAK*
Robert SMYK*
Zenon ULMAN*

PIPELINED SCALING OF SIGNED RESIDUE NUMBERS WITH THE MIXED-RADIX CONVERSION IN THE PROGRAMMABLE GATE ARRAY

In this work a scaling technique of signed residue numbers is proposed. The method is based on conversion to the Mixed-Radix System(*MRS*) adapted for the *FPGA* implementation. The scaling factor is assumed to be a moduli product from the Residue Number System (*RNS*) base. Scaling is performed by scaling of terms of the mixed-radix expansion, generation of residue representations of scaled terms, binary addition of these representations and generation of residues for all moduli. The sign is detected on the basis of the value of the most significant coefficient of the *MRS* representation. For negative numbers their residues are adequately corrected. The basic blocks of the scaler are realized in the form of the modified two-operand modulo adders with included additional multiply and modulo reduction operations. The pipelined realization of the scaler in the Xilinx environment is shown and analyzed with respect to hardware amount and maximum pipelining frequency. The design is based on the *LUT*s(2^6x 1) that simulate small *RAM*s serving as the main component for the look-up realization.

1. INTRODUCTION

The Residue Number System (*RNS*) [1],[2],[3] is a non-positional number system that has been invented in order to decompose certain operations on large integers into sets of operations on small numbers. The simplicity of such *RNS* operations as addition, subtraction and multiplication is offset by the difficult realization of scaling, division, sign detection, magnitude comparison and overflow detection. Hence the *RNS* can be advantageous for the realization of these algorithms where the operations of the first group dominate. To such algorithms belong those of the digital signal processing such as the Finite Impulse Response (*FIR*) and the Fast Fourier Transform (*FFT*). In these algorithms the multiplications represent the most expensive operation, however when the coefficients of the algorithm are fixed, the more simple multiplication by a

* Gdańsk University of Technology.

constant can be applied. In the *RNS* the multiplication is decomposed into a set of multiplications of small numbers with the binary size of the *RNS* moduli. For the 5-bit moduli the multiplications by a constant can be performed by look-up using logic functions of five variables with 5-bit look-up tables. The latest Field Programmable Gate Arrays (*FPGA*) as Virtex-7 by Xilinx [4] have 6-bit *LUT* address, hence they can be used directly for 6-bit moduli. The multiplication problem has become much easier for the applications that can be implemented on the *FPGA* platform due to the introduction of the fast Multiply-Accumulate (*MAC*) units (*DSP* slices) that comprise 18x25 bit multipliers. For example, Virtex-7 XC7VH870T contains 1920 *DSP* slices, whereas the total number of 6-bit *LUT*'s is 547600.

As the *RNS* is an integer number system, the fractional or real coefficients of the *DSP* algorithm have to be transformed to integers by multiplication by a suitable constant, *K* and appropriate rounding off. *K* should be large enough to provide for the required accuracy of the representation of algorithm coefficients. This transformation makes that the sum of products of input signal samples and the transformed coefficients becomes close to the *RNS* number range, *M*. In order to avoid overflow in the next processing stage, the sum represented in the *RNS* has to be divided by *K* or by the number comprising also the growth of the dynamic range of the signal resulting from summation of terms.

The scaling algorithms were presented in several works[1], [5-12]. The main difficulty was the need to use memories that reduced the processing speed. The algorithm by Szabo and Tanaka[1] allowed for scaling by a product moduli from the *RNS* base in *n* clock cycles, where the clock cycle denotes the time required for the elementary operation such as residue multiplication by a constant or residue subtraction. Jullien [5] proposed an algorithm that permitted for scaling by a product of *s* moduli out of the *n RNS* moduli in $n+\lceil \log_2 s+1 \rceil$cycles. The main group of scaling techniques are the methods based on the Mixed-Radix System (*MRS*) or the *CRT* decomposition.There is also a method based on core function [15].Taylor and Huang [6] presented a technique called the autoscale multiplier, where the scaling process is performed by truncating the mixed-radix conversion to a level, where the binary size of the scaled integer does not exceed the binary size of the address of high-speed memories. Miller and Polky [7] reported a scaling technique in *n*+1 clock cycles with an absolute error bound of$(n-i_0/2)$, wherei_0 is an index, $i_0 \le n$ used in rounding the scaled integer, *i.e.*, only i_0 mixed-radix coefficients are used for the representation of the scaled integer. The scaling using the *CRT* was attempted by Jenkins [8], but this technique was inefficient due the necessity of performing the modulo *M* operation without having the suitable fast algorithms. Jullien [5] also described a technique based on "estimates" and the *CRT* with the possibility of emerging of large error. Shenoy and Kumaresan [9] described a method that allows scaling by a product

of moduli in *log n* cycles, where the cycle denotes the access time of the high speed memory. The technique uses the *CRT* and the redundant modulus to compute the magnitude index in the *CRT* formula. Ulman and Czyżak [10] proposed a scaling technique in non-redundant residue arithmetic that uses only small memories with the size at most of $\lceil \log_2 m \rceil + 1$ -bits, and arithmetic elements, where $\lceil \log_2 m \rceil$ is the binary size of the modulus. In this technique, *K* can be a real number. A novel concept of scaling was recently presented by Meyer-Baese and Stouraitis [11], they proposed effective scaling by 2 by transforming the scaling operation into division remainder zero by checking parity of the number and adding 1 when necessary in order to assure the existence of multiplicative inverses of 2 with respect to all moduli of the *RNS* base. This method can be extended by repeating scaling by 2, or by directly using the power of 2, but this approach requires larger look-up tables. The known scaling methods have certain drawbacks that makes their application in the high-speed *DSP* difficult. The first drawback is the special form of the moduli of the *RNS* base and their fixed and limited number, that may enforce their increased size to attain the necessary dynamic range. The increased size makes other operations like multiplication by a constant not realizable by table look-up. The second is the use of large look-up tables, that practically excludes pipelining. The third is the limitation imposed on the form of the scaling factor. The scaling factor is usually limited to one or two moduli or their product. Moreover, the majority of the known methods do not provide the scaling of signed numbers with the implicit sign. The certain remedy may be the use of scaling techniques termed the approximate *CRT* methods. They allow to reduce the scaler complexity and provide other desirable characteristics. Griffin et. *al*. [12] presented a method termed (L+δ)-*CRT*]. The scaling factor can be any number from [0, *M*). This technique allows to use approximate scaled projections and also instead modulo *M* operation, operation modulo μ, where μ is a more convenient number with respect to modulo reduction. However, the use of the approximate values may lead to large, unacceptable errors in scaling of signed numbers.

In this work a scaling technique of signed residue numbers with implicit sign based on the *MRC* for *DSP* applications is presented. It is assumed the *RNS* base consists of five-bit moduli. Such moduli provide for the necessary dynamic range in many applications and also they allow to easily implement modulo multiplication by a constant in the *FPGA* environment. In the paper first the *RNS*, *MRC* and fundamental issues of scaling are reviewed, next the conversion of the *MRC* algorithm to the form suitable for implementation the *FPGA* enrironment, the scaling technique and the scaler architecture are described. Finally the results of implementation are shown.

2. THE RNS, MRS AND SCALING FUNDAMENTALS

A. Residue Number System

The nonnegative integer N from the number range $[0, M-1]$ is represented in the *RNS* by the digit vector $\left(|N|_{m_1}, |N|_{m_2}, \ldots, |N|_{m_n}\right)$, where $|N|_{m_j}$ is the least nonnegative residue from the division of N by m_j, $j = 1,2,3,\ldots,n$. The numbers m_j, termed the moduli, are the elements of the system base, $B = \{m_1, m_2, \ldots, m_n\}$ and $M = \prod_{j=1}^{n} m_j$. If the moduli m_j are pair-wise mutually prime, there is one-to-one mapping between the number set and the representation set, given by the Chinese Remainder Theorem, where the value of an integer N is given by the formula

$$N = \left| \sum_{j=1}^{n} N_j \right|_M , \tag{1}$$

with $N_j = M_j \cdot \left| M_j^{-1} \cdot |N|_{m_j} \right|_{m_j}$, $M_j = M/m_j$, and $\left| M_j \cdot M_j^{-1} \right|_{m_j} = 1$. M_j^{-1} is the multiplicative inverse of M_j modulo m_j, and exists if $gcd\left(M_j, m_j\right) = 1$. For signed numbers denoted as X, if M is even, $X = N$ for $N < M/2$, and $X = N - M$. if $N \geq M/2$. If M is odd, $X = N$ for $N < (M-1)/2$, and $X = N - M$. if $N \geq (M-1)/2$.

B. The Mixed Radix System(MRS)

X with the *RNS* representation $\left(|X|_{m_1}, |X|_{m_2}, \ldots, |X|_{m_n}\right)$ can be represented in the *MRS* as

$$X = \sum_{i=1}^{n} a_i \prod_{j=1}^{i-1} m_j \tag{2}$$

where

$$a_1 = |X|_{m_1} \tag{3a}$$

$$a_2 = \left| |X - a_1|_{m_2} \cdot \left| \frac{1}{m_1} \right|_{m_2} \right|_{m_2} \tag{3b}$$

$$a_3 = \left| \left| X - a_1 - a_2 \right|_{m_3} \cdot \left| \frac{1}{m_1 \cdot m_2} \right|_{m_3} \right|_{m_3} \tag{3c}$$

$$a_i = \left| \left| X - \sum_{j=1}^{i-1} a_i \cdot \prod_{k=1}^{i-2} m_k \right|_{m_3} \cdot \left| \frac{1}{\prod_{j=1}^{i-1} m_j} \right|_{m_{ii}} \right|_{m_i} ,i=4,..,n. \tag{3d}$$

The *RNS*/*MRS* conversion by (3) has a sequential form, but also exists the parallel methods. The important advantage of the *MRS* is sign detection, especially easy if $|m_n|_2 = 0$. If $a_n < m_n / 2$ then $X \geq 0$ else X<0.

B. Scaling fundamentals

We first consider scaling of unsigned numbers. *i.e.* $X = N$. If K is a positive integer scaling factor, the scaling result, Y has the following form

$$Y = \frac{X - |X|_K}{K}. \tag{4}$$

Provided that $|1/K|_{m_i}$ exists, the corresponding residue of Y, y_i is given by

$$y_i = |Y|_{m_i} = \left| (x_i - \left| |X|_K \right|_{m_i}) \cdot \left| \frac{1}{K} \right|_{m_i}) \right|_{m_i}. \tag{5}$$

If K is a real number, the residues of the scaling result can be expressed as

$$|Y|_{m_i} = \left| \left\{ \frac{X}{K} \right\} \right|_{m_i} ,i=1,2,..,n, \tag{6}$$

where $\{\cdot\}$ denotes the rounding off to the nearest imteger.

$|Y|_{m_i}$ can be determined in (5) for those moduli for which $|1/K|_{m_i}$ exists, this is the case when m_i, i=1,2,...,n and K are mutually prime. K may be chosen as a product of certain moduli of B. Assume without of loss of generality, that first s moduli of B are chosen, *i.e.*,

$$K = \prod_{i=1}^{s} m_i \tag{7}$$

with $s<n$. Then the multiplicative inverse of K for m_i ,$i=1,2,...,s$ does not exist and y_i for these moduli cannot be computed by using (5). In [13] Garcia and Lloris proposed the calculation of y_j , $j=s+1,...,n$, with (5) by look-up with the use of $y_1, y_2,..., y_s, y_s$ as the memory address . This technique may require large look-up tables that excludes low-level pipelining. The second problem in (5) is the computation of $|X|_K$. If K is equal to one of the moduli no computation is needed, or its also simple if the binary size of K does not exceed the acceptable length of the look-up table address. Generally the computation of $|X|_K$ requires the conversion to a weighted system, for example, to the *MRS*. The *MRS* also facilitates scaling if K is a product of the *RNS* moduli,

The scaling can be also performed by using the conversion to the mixed-radix form, division of each term by K and rounding off the quotient.. If we represent X in the mixed-radix system

$$X = a_n m_1 m_2 ... m_{n-1} + ... + a_2 m_1 + a_1 , \tag{8}$$

where $0 \le a_j < m_j$. after division of each term by K, we obtain

$$Y = \frac{a_n m_1 m_2 ... m_{n-1}}{K} ... + \frac{a_2 m_1}{K} + \frac{a_1}{K} \tag{9}$$

After rounding off the individual quotient to integers, we may obtain the residues of Y,

$$Y = \left| \left\{ \frac{a_1 m_1 m_2 ... m_{n-1}}{K} \right\} ... + \left| \left\{ \frac{a_2 m_1}{K} \right\} \right|_{m_i} + \left| \left\{ \frac{a_1}{K} \right\} \right|_m \right|_{m_i} \tag{10}$$

where $\{\cdot\}$ denotes rounding off to the nearest integer. When K is the product of first s moduli of the *RNS* base the scaling result has the following form

$$|Y|_{m_i} = \left| \left| \sum_{i=1}^{s} \frac{a_i}{\prod_{k=i}^{s+1-k} m_k} \right|_{m_i} + |a_{s+1}|_{m_i} + \left| \sum_{i=s+2}^{n-s-1} a_i \cdot \prod_{k=s+1}^{n-1} m_k \right|_{m_i} \right|_{m_i} \tag{11}$$

Using (10), scaling can be performed by the summation of the rounded quotients and the forward *RNS* conversion. The error will not exceed $0.5n$, but it can be reduced by applying the additional error compensation channel that will sum up the properly rounded fractional parts of the quotients. When K is a moduli product, error compensation under certain conditions can be avoided. For example, let $n = 6$, then

$$X = a_6 m_1 m_2 m_3 m_4 m_5 + a_5 m_1 m_2 m_3 m_4 + a_4 m_1 m_2 m_3 + a_3 m_1 m_2 + a_2 m_1 + a_1 , \quad (12)$$

moreover, assume $K = m_1 m_2 m_3 m_4$. We get

$$Y_R = X / K = a_6 m_5 + a_5 + \frac{a_4}{m_4} + \frac{a_3}{m_3 m_4} + \frac{a_2}{m_2 m_3 m_4} + \frac{a_1}{m_1 m_2 m_3 m_4} , \quad (13)$$

We can truncate fractional terms in (13) and the maximum error will be equal to

$$\varepsilon_t = \frac{m_4 m_3 m_2 m_1 - 1}{m_4 m_3 m_2 m_1} . \quad (14)$$

The integer scaling result by (14) is as follows

$$|Y|_{m_i} = |a_6 m_5 + a_4|_{m_i} \qquad i=1,2,3,4,5,6. \quad (15)$$

Operations in (15) can be performed with one look-up and two-operand modulo m_i addition. If the number of terms is greater than 2, multi-operand modulo addition is needed. .

C .Scaling of signed numbers with the implicit sign

The scaling of signed integers with the implicit sign can be carried out in the manner given in[14]. Let N_Y covers the interval $[0, M / K)$. The scaling result, *Y*, obtained by scaling of a signed integer X , is determined as follows

For $|M/K|_2 = |M|_{2K} = 0$,

$$Y = N_Y \qquad \text{if } N_Y < M / 2K , \quad (16a)$$

or

$$Y = N_Y - M/K \qquad \text{if } N_Y \geq M / 2K . \quad (16b)$$

For $|M / K|_2 = |M|_{2K} \neq 0$,

$$Y = N_Y \qquad \text{if } N_Y \leq (\lfloor M / K \rfloor - 1)/2 \quad (17a)$$

or

$$Y = N_Y - \lfloor M/K \rfloor \qquad \text{if } N_Y \geq (\lfloor M / K \rfloor - 1)/2 . \quad (17b)$$

Hence for (16) we receive the residues of the scaling result in the *M* ring as

$$|Y|_{m_i} = \left| M - \frac{M}{K} + N_y \right|_{m_i} = m_i - \left| \frac{M}{K} - N_Y \right|_{m_i} , \; i=1,2,...,n. \quad (18)$$

Next we shall consider the computation of (18) with the use of the *MRS*.

Example 1. Scaling of negative number using the *MRC* for $B = \{m_1, m_2, m_3, m_4\}$ and $K = m_1 m_2 m_3 m_4$. N_Y can expressed as

$$N_Y = \frac{N}{K} = a_4 m_3 + a_3 + \frac{a_2}{m_1} + \frac{a_1}{m_1 m_2}. \tag{19}$$

After truncation we receive for nonnegative numbers

$$|Y|_{m_i} = \lfloor N_Y \rfloor = a_4 m_3 + a_3, \tag{20}$$

and for negative

$$|Y|_{m_i} = m_i - |m_3 m_4 - a_4 m_3 - a_3|_{m_i}, \; i=1,2,...,4. \tag{21}$$

3. SCALER ARCHITECTURE AND IMPLEMENTATION

In Fig. 1 the scaler architecture is shown for the base $B = \{m_1, m_2, m_3, m_4\} = \{27,29,31,32\}$ and $K = m_1 m_2 = 27 \cdot 29 = 783$. It is assumed that all moduli have 5-bit binary length. We first describe the implementation of the *RNS/MRS* converter. The computations are performed by using (22-25).

$$a_1 = |X|_{m_1} \tag{22}$$

$$a_2 = \left| \left| |X|_{m_1} - a_1 \right|_{m_2} \cdot \left| \frac{1}{m_1} \right| \right|_{m_2} \tag{23}$$

$$a_3 = \left| \left(|X|_{m_3} - a_1 \right) + \left| m_3 - |a_2 m_1|_{m_3} \right|_{m_3} \cdot \left| \frac{1}{m_1 m_2} \right|_{m_3} \right|_{m_3} \tag{24}$$

$$a_4 = \left| \left(|X|_{m_4} - a_1 \right) + \left| m_2 - |a_2 m_1|_{m_4} \right|_{m_4} + \left| m_3 - |a_3 m_1 m_2|_{m_4} \right|_{m_4} \cdot \left| \frac{1}{m_1 m_2 m_3} \right|_{m_4} \right|_{m_4} \tag{25}$$

In the first stage the binary adders *BA11-BA13* compute the differences $|X|_{m_i} - a_i$, $i = 1,2,3,4$.(the first digit in the description of the blocks denotes the number of the stage). It is assumed that a_1 is two's complement encoded. *ROM*11 performs the multiplication of the difference $|X|_{m_1} - a_1$ by the multiplicative inverse of m_1 and reduction of the product modulo m_2. For computation of a_3 first the difference $\left(|X|_{m_3} - a_1 \right)$ is calculated by the binary adder *BA*32 and next $m_2 - |a_2 m_1|_{m_4}$ by *ROM*12. *BA*21 adds the outputs of *ROM*12 and *ROM*13 and multiplies the result by the multiplicative inverse $\left| \frac{1}{m_1 m_2} \right|_{m_3}$. Next *ROM*21 computes a_3. In the final converter stage a_4 is determined with the use of *BA*31

and *ROM*31. The *ROM*s in the 4th stage calculate the residues of the scaling result by (20). *ROM*51-*ROM*54 perform $m_i - y_i$. The sign of the *RNS* number is detected on the basis of a_4 value. The sign of the number controls the multiplexer and decides whether the result is computed by (20) or (21). The scaler shown in Fig. 1 has been implemented in Xilinx *FPGA* environment. The following implementation results have been obtained.

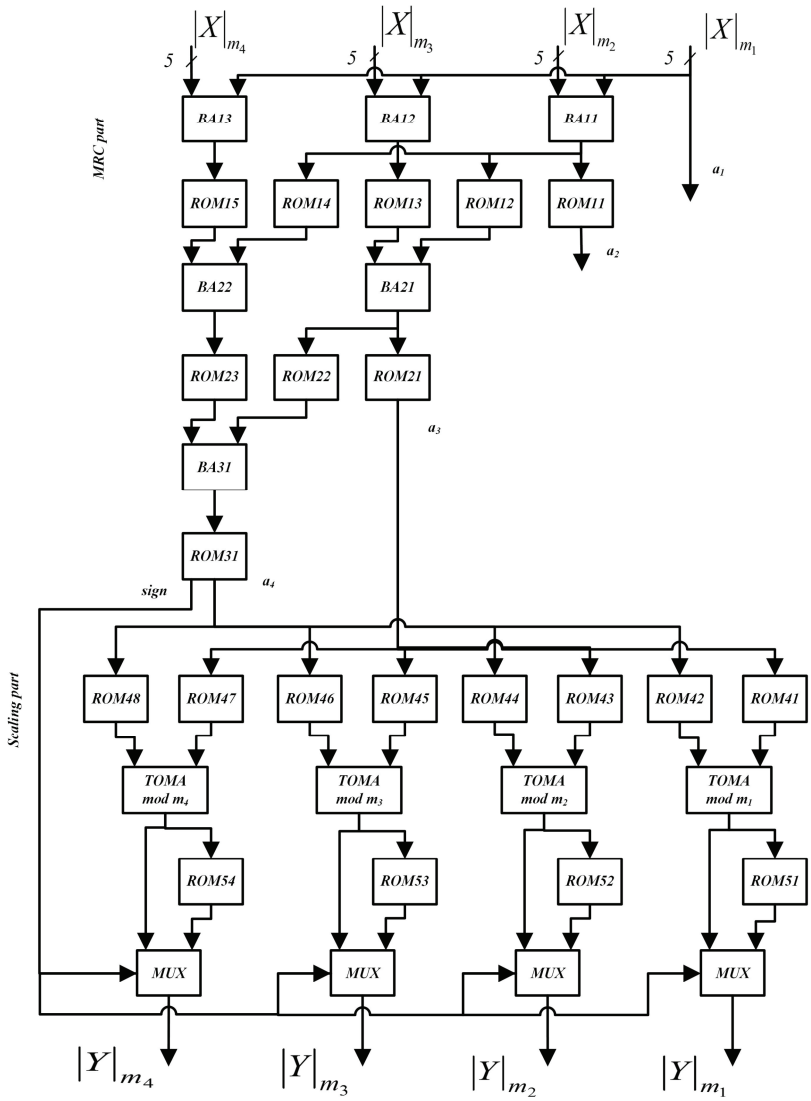

Fig. 1. The architecture of the residue scaler of signed numbers based on the *MRS*

```
Device utilization summary:
Selected Device : 6vcx240tff784-2

Slice logic utilization:
   Number of slice registers:             106  out of  301440
   Number of slice LUTs:                  119  out of  150720    0%
   Number used as logic:                   99  out of  150720     0%
    Number used as memory:                 20  out of  58400      0%

       Timing Summary:
---------------
Speed Grade: -2
Minimum period: 1.518ns (maximum frequency: 658.610MHz)
   Minimum input arrival time before clock: 0.550ns
   Maximum output required time after clock: 0.659ns
```

It is seen the that high pipelining frequency can be attained. It can also be remarked that the scaler occupies only a small portion of the available resources of the given device.

4. CONCLUSIONS

The paper presents an approach to scaling of signed residue numbers based on the use of mixed-radix conversion. The signed residue number is converted to the mixed-radix system, in parallel with the computation of the *MRS* highest coefficient, the sign detection is performed. The scaling factor is a product of moduli of the *RNS* base, that simplifies scaling. Scaling is performed by the division of the individual terms of the *MRS*, conversion of the individual quotients to the *RNS*, and modulo addition in every residue channel. The proper residue of the scaling result is obtained by multiplexing of the residue received for nonnegative and negative numbers. The scaler architecture utilizes the small *ROM*s simulated by *LUT*s in the Xilinx Virtex-6 architecture. The presented architecture makes use of the 4-moduli base and small scaling factor but it can be easily extended to more practical 6- or 7 moduli *RNS* base.

REFERENCES

[1] Szabo N.S, R.J., Tanaka R.J, *Residue Arithmetic and its Applications to Computer Technology*, New York, McGraw-Hill, 1967.
[2] Soderstrand M. *et al.*, *Residue Number System Arithmetic*: Modern Applications in Digital Signal Processing, IEEE Press, NY, 1986.
[3] Omondi A., Premkumar B.: *Residue Number Systems: Theory and Implementation,* London, Imperial College Press, 2007.
[4] Xilinx, Virtex-7, www.xilinx.com.

[5] Jullien G.A.: Residue number scaling and other operations using ROM arrays, IEEE Trans. on Computers, Volume C-27, pp. 325-336, April 1978.
[6] Taylor F.J., Huang C.H.: An autoscale residue multiplier, IEEE Trans. on Computers, Volume C-31, Number 4, pp. 321-325, April 1982.
[7] Miller D.D., Polky J.N.: An implementation of the LMS algorithm in the residue number system, IEEE Trans. on Circuits and Syst., Volume 31, pp.452-461. May 1984.
[8] Jenkins W.: Recent advances in residue number techniques for recursive digital filtering, IEEE Trans. on Acoust., Speech and Signal Processing, Volume 27, Number 1, pp. 19-30, Feb. 1979.
[9] Shenoy M.A.P. , Kumaresan R.: A fast and accurate RNS scaling technique for high-speed signal processing, IEEE Trans. Acoust. Speech, Signal Processing, Volume 37, pp. 929-937, June 1989.
[10] Ulman Z.D., Czyżak M.: Highly parallel fast scaling of numbers in nonredundant residue arithmetic, IEEE Trans. on Signal Processing, Volume 46, pp.487-496, Feb. 1998.
[11] Meyer-Baese U., Stouraitis T.: New power-of-2 RNS scaling scheme for cell based IC design, IEEE Trans. on Large Scale Integration(VLSI) Systems, Volume 11, Number 2, pp. 280-283.
[12] Griffin M., Sousa M., Taylor F.: Efficient scaling in the residue number system, Proc. ASSP'98, pp. 1075-1078.
[13] Garcia A, Lloris A.: A look-up scheme for scaling in the RNS, IEEE Trans. on Computers, Volume 48, Number 7, pp. 748-751, July 1999.
[14] Ulman Z., Czyżak M.: Scaling of numbers with implicit sign in residue arithmetic, XIXth National Conf. Circuit Theory and Electronic Networks, Volume 2, pp. 505-510, 1996.
[15] Burgess N.: Scaling an RNS Number Using the Core Function, 16th IEEE Symposium on Computer Arithmetic 2003, Santiago de Compostela, Spain, pp. 262-269, 2003.