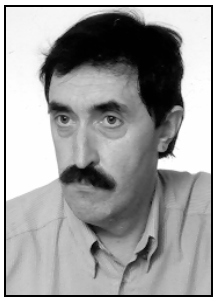
Maciej CZYŻAK
POLITECHNIKA GDAŃSKA, WYDZIAŁ ELEKTROTECHNIKI

An improved high-speed residue-to-binary converter based on the Chinese Remainder Theorem

Ph.D. Maciej CZYŻAK

He received his Ph.D. in 1985 from the Gdansk University of Technology(GUT). From 1984 to 1990 he was a researcher and lecturer at the Institute of Informatics, Faculty of Electronics, GUT. Since 1994 he has been a lecturer of informatics at the Faculty of Electrical and Control Engineering ,GUT. His research interests are in fast digital signal processing, computer arithmetic and VLSI design.



e-mail: m.czyzak@ely.pg.gda.pl

Abstract

A new high-speed residue-to-binary converter for five bit moduli based on the Chinese Remainder Theorem is presented. The orthogonal projections are computed by mapping using five-variable logic functions. The sum of projections is calculated using the Wallace tree. The output carry-save representation is partitioned into four segments in such a way that the sum of the numbers represented by the low-order segments does not exceed the Residue Number System (RNS) range *M*. The bits of the high-order segments are compressed by the small carry-propagate adder that in effect diminishes the size of the modulo *M* generator used to reduce the number represented by the high-order segments. The obtained sum is smaller than 2*M*, thus the effective two-operand final modulo *M* adder can be used. The proposed converter can be pipelined on the full-adder level.

Keywords: residue number system, residue-to-binary conversion.

Ulepszony szybki konwerter z systemu resztowego do systemu binarnego oparty na chińskim twierdzeniu o resztach

Streszczenie

Zaprezentowano nową architekturę konwertera z systemu resztowego do systemu binarnego dla modułów 5-bitowych opartą na chińskim twierdzeniu o resztach. Projekcje ortogonalne określane są poprzez odczyt z pamięci ich obliczonych wartości. Pamięć symulowana jest poprzez użycie funkcji logicznych o liczbie zmiennych równej bitowej długości modułu. Suma projekcji obliczana jest przy użyciu sumatora wielooperandowego opartego na drzewie Wallace'a. Wyjściowe wektory sumy i przeniesienia są dzielone na cztery segmenty w taki sposób, że suma liczb reprezentowanych przez bity o młodszych wagach nie przekracza zakresu liczbowego systemu resztowego, *M*. Bity należące do segmentów o starszych wagach są dodawane w niewielkim sumatorze, co w efekcie umożliwia znaczne zmniejszenie rozmiaru generatora stosowanego do redukcji modulo *M* liczby reprezentowanej przez te bity. Suma otrzymana po zsumowaniu liczby reprezentowanej przez segmenty o młodszych wagach i zredukowanej liczby reprezentowanej przez segmenty starszych wagach nie przekracza 2*M*, co umożliwia zastosowanie efektywnego sumatora końcowego modulo *M*. Konwerter w proponowanej konfiguracji może pracować potokowo na poziomie pełnego sumatora.

Słowa kluczowe: system resztowy, konwersja z systemu resztowego do systemu binarnego.

1. Introduction

The digital processing of signals in real time requires high-speed architectures. The Residue Number System (RNS) [1] is an effective tool for these applications where add-multiply operations dominate since it permits high-speed, low-level pipelined realization of addition, subtraction and multiplication. The RNS replaces operations in a large integer ring by the set of operations in smaller integer rings. Addition, subtraction and multiplication are performed independently on the corresponding digits of the representations of processed numbers without carries between the positions. In the RNS, the nonnegative integer X from the number range $[0, M\text{-}1]$ is represented by n-tuple $\left(|X|_{m_1}, |X|_{m_2}, ..., |X|_{m_n}\right)$, where the digit $|X|_{m_j}$ is the smallest nonnegative residue from the division of X by m_j, $j = 1, 2, 3, \ldots, n$, where m_j are the elements of the system base, $B = \{m_1, m_2, ..., m_n\}$ and $M = \prod_{j=1}^{n} m_j$. The moduli m_j should be pair-wise co-prime in order to preserve one-to-one mapping between the number set and the representation set. The relationship between these sets is given by the Chinese Remainder Theorem (CRT)[1]. Each RNS-based architecture requires conversions from the binary system to the RNS (forward conversion) and from the RNS to the binary system (reverse conversion). The complexity of the reverse conversion along with the complex operations of sign detection, magnitude comparison, scaling and division, have limited the use of the RNS to the situations where it seems to be the only possible solution. The reverse conversion has been implemented using the Mixed-Radix Conversion(MRC) algorithm [1-5], or the CRT [1, 2], [6-12], or the concurrent use of the CRT and MRC [13] or the core function [14]. Recently algorithms have emerged [9], where the conversion formula has a new form. The MRC process is relatively slow, and therefore the CRT-based conversion is currently the most frequent approach. The main disadvantage is the modulo M operation with M being a large number. In principle, there are two effective approaches. The first was proposed by Elleithy and Bayoumi [7], where they suggested an effective structure using a range determinator, so that the appriopriate value can be subtracted using the carry-look ahead adder. The second approach was introduced by Piestrak [8], where an efficient reduction of the Wallace tree output to the interval [0, 2*M*) is shown by using the modulo *M* generator to reduce the number represented by the most significant bits of the carry and save vectors. This simplifies the final modulo *M* generation. Such conversion technique uses only one $\lceil \log_2 M \rceil$-bit carry-propagate addition in series. However, for a greater number of moduli this approach becomes less effective due to the growth of the ROM used in the structure. In this work a new converter structure is proposed being an extension of the technique from [8]. Two important modifications are introduced. One is due to the replacement of all ROM's by effectively implemented five-variable logic functions. The other is the compression, by using a small binary adder, of the most significant bits(msb) of the carry-save representation prior to mapping. This reduces the size of the look-up table needed in the modulo *M* generation which makes possible a memoryless realization of the converter. The time-hardware complexity of the new architecture is comparable with other known designs, but the pipelining rate can potentially be on the Full Adder (FA) level. The time-hardware complexity is analyzed using the standard cell library[17]. This paper is the revised a version of the material initially presented in [18].

2. New CRT computation

The value of an integer X, with the use of the CRT[1], is given by the formula

$$X = \left| \sum_{j=1}^{n} X_j \right|_M , \qquad (1)$$

with $X_j = M_j \cdot \left| M_j^{-1} \cdot |X|_{m_j} \right|_{m_j}$, $M_j = M/m_j$, and $\left| M_j \cdot M_j^{-1} \right|_{m_j} = 1$.

M_j^{-1} is called the multiplicative inverse of M_j modulo m_j, and exists if $\gcd(M_j, m_j) = 1$. The use of (1) requires the initial computation of the projections and a multi-operand adder modulo M, where M is usually large. The adder itself, a complex structure, based on n-1 two-operand modulo M adders, it is not considered here. The alternative solution can be based on the Wallace tree and subsequent modulo M reduction. This reduction can be made by first doing the binary carry-propagate addition, but in this case a large binary adder is necessary. One technique to avoid it, is to partition the carry-save representation in such a way that the low-order segments of carry and save vectors represent jointly a number smaller than M [8]. Then after the modulo M reduction of the number represented by the high-order segments, the obtained sum does not exceed 2M. Below we shortly present the modified version of this procedure. The sum of projections at the Wallace tree output can be written as

$$\left| \sum_{j=1}^{n} X_j \right|_M = C^{(1)} + S^{(1)}, \tag{2}$$

where $C^{(1)}$ and $S^{(1)}$ are the numbers represented by the carry and save vectors, respectively. After partitioning we get

$$C^{(1)} = C_H + C_L \tag{3a}$$

and

$$S^{(1)} = S_H + S_L, \tag{3b}$$

where C_H and S_H are the numbers represented by the high-order bits of $C^{(1)}$ and $S^{(1)}$, while C_L, S_L by the low-order bits. The partitions in (3) are done so that

$$C_L + S_L < M, \tag{4}$$

that will allow to limit the processed sum to [0,2M). We assume here that

$$\lceil \log_2 C_L \rceil = \lceil \log_2 S_L \rceil = \lceil \log_2 M \rceil - 2. \tag{5}$$

Next we do the compression of msb's

$$S^{(2)} = C_H + S_H, \tag{6}$$

and perform the modulo M reduction by mapping

$$S^{(3)} = \left| S^{(2)} \right|_M, \tag{7}$$

and subsequently the following addition is carried out

$$S = S^{(3)} + C_L + S_L. \tag{8}$$

Then X can be evaluated as

$$X = S - M \text{ if } S \geq M, \tag{9a}$$

or

$$X = S \text{ if } S < M. \tag{9b}$$

The efficiency of this procedure is due to addition in (6), that allows the modulo M operation in (7) to be done by look-up implemented with logic functions of at most six variables for $n \leq 16$. This is because for such n, there is $\lceil \log_2 n \cdot M \rceil - \lceil \log_2 M \rceil - 2 \leq 6$.

3. The conversion algorithm

Given below is a step-by-step specification of the residue-to-binary conversion algorithm.

Input: The residue number X, represented as

$$X = (x_n, x_{n-1}, ..., x_1).$$

Output: The binary representation of X.

Step 1: Compute in parallel the orthogonal projections X_j for the individual x_j, j=1,..., n.

$$X_j = \left| \left| x_j \cdot N_j \right|_{m_j} \cdot M_j \right|_M \tag{10}$$

Step 2: Compute

$$\sum_{j=1}^{n} X_j = S^{(1)} + C^{(1)} =, \tag{11}$$

using the Wallace tree.

Step 3: Partition the Wallace tree output vectors

$$S = (s_{k_1}, s_{k_1 - 1}, ..., s_0) \text{ and } C = (c_{k_2}, c_{k_2 - 1}, ..., c_0)$$

into segments

$$S_H = \left(s_{k_1}^{(H)}, s_{k_1 - 1}^{(H)}, ..., s_{k_3}^{(H)} \right), \tag{12a}$$

$$S_L = \left(s_{k_3 - 1}^{(L)}, s_{k_3 - 2}^{(L)}, ..., s_0^{(L)} \right), \tag{12b}$$

with

$$k_3 = k_4 = \lceil \log_2 M \rceil - 2,$$

$$C_H = \left(c_{k_2}^{(H)}, c_{k_2 - 1}^{(H)}, ..., c_{k_4}^{(H)} \right), \tag{12c}$$

$$C_L = \left(c_{k_4 - 1}^{(L)}, c_{k_4 - 2}^{(L)}, ..., c_0^{(L)} \right), \tag{12d}$$

so that the inequality is fulfilled

$$\sum_{i=1}^{k_3 - 1} 2^i + \sum_{j=0}^{k_4 - 1} 2^j < M. \tag{13}$$

Step 4: Compute

$$S^{(2)} = \sum_{i=k_4}^{k_2} c_i^{(L)} 2^i + \sum_{i=k_3}^{k_1} s_i^{(L)} 2^i, \tag{14}$$

using the carry-propagate adder of the length

$$l_{CPA} = \lceil \log_2 nM \rceil - \lceil \log_2 M \rceil + 1.$$

Step 5: Compute by mapping

$$S^{(2)} \leftrightarrow (s_v, s_{v-1}, ..., s_0) \rightarrow \left| S^{(2)} \right|_M = S^{(3)}. \tag{15}$$

Step 6: Compute

$$S^{(4)} + C^{(4)} = S^{(3)} + \sum_{i=0}^{k_4 - 1} c_i^{(L)} 2^i + \sum_{i=0}^{k_3 - 1} s_i^{(L)} 2^i, \tag{16}$$

using the CSA adder.

Step 7. Subtract M using two's complement

$$S = S^{(5)} + C^{(5)} = S^{(4)} + C^{(4)} + 2^{\lceil \log_2 M \rceil} - M \tag{17}$$

using the CSA adder.

Step 8. Compute in parallel

$$S^{(6)} = S^{(5)} + C^{(5)} \tag{18a}$$

and

$$S^{(7)} = S^{(4)} + C^{(4)}. \tag{18b}$$

using two CPA adders.

If $S^{(6)} \geq 2^{\lceil \log_2 M \rceil}$ set $X = S^{(6)}$ else set $X = S^{(7)}$.

4. Architecture of the new converter

The architecture of the new converter that implements the consecutive steps of the above algorithm is shown in Fig. 1. In the first stage the look-up tables $LT_1^{(q)},...,LT^{(q)}{}_N$ are used, implemented as the $\lceil \lceil \log_2 M \rceil / \lceil \log_2 m \rceil \rceil$ logic blocks composed of q-variable $\lceil \log_2 m \rceil$ logic functions. (here q=5). The residues are added by n -operand Wallace tree (CSA1). The high-order segments are added by CPA1 and next the obtained sum is reduced modulo M by using the look-up table $LT_{N+1}^{(q)}$. The CSA2 implements carry-save addition of the number represented by low-order segments and the number obtained from the modulo M reduction of high-order segments. The CSA3 makes the subtraction of M in two's complement. The CPA2 performs binary addition and the CPA3 two's complement addition. Subsequently the output carry from the CPA3 is used to select the correct result by using the multiplexer (MUX1). The placement of latches is not shown in Fig. 1. In order to attain pipelining on the FA level, the latches have to be placed inside the logic blocks, since their delay is twice the FA delay.

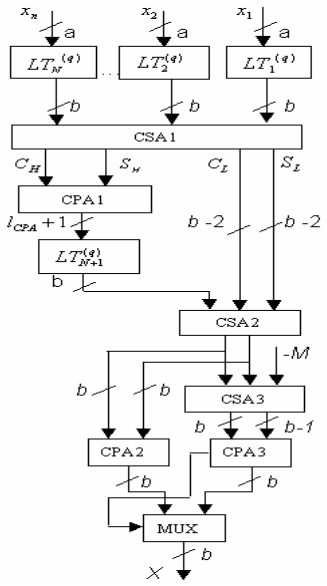


Fig. 1. The architecture of the new converter ($b = \lceil \log_2 M \rceil$, $a = \lceil \log_2 m \rceil$, $l_{CPA} = \lceil \log_2 nM \rceil - \lceil \log_2 M \rceil + 1$)

Rys. 1. Architektura nowego konwertera ($b = \lceil \log_2 M \rceil$, $a = \lceil \log_2 m \rceil$, $l_{CPA} = \lceil \log_2 nM \rceil - \lceil \log_2 M \rceil + 1$)

5. Time-hardware complexity of the new converter

The method of evaluation of time-hardware complexity depends on purpose of evaluation and the type of a VLSI circuit. For evaluation of ASIC architectures based on standard cell library(STCL), a rough estimate of the circuit area can be obtained by summing the area of the required logic elements, expressed in μm^2 or GE (Gate Equivalents), where the GE is the area of a two-input NAND with fan-out=1. However, the use of the FA area is more common. Also the FA delay t_{FA} is used as the unit. The STCL also allows a rough estimate of power consumption for the required frequency of operation. The models of logic components used here are from Samsung 0.18μ STDL 130 [17].

In the next subsections we shall analyze the hardware amount (HA) and delay of the new converter.

5.1. Hardware amount

The hardware amount (HA) of the converter architecture shown in Fig. 1 can be estimated in the following manner

$$\begin{aligned} A_{CONV} = n \cdot \lfloor \lceil \log_2 M \rceil / \lceil \log_2 m \rceil A_{LF(q,\lceil \log_2 m \rceil)} \rfloor + A_{CSA(n,\lceil \log_2 M \rceil)} + \\ + \lceil \lceil \log_2 M \rceil / \lceil \log_2 m \rceil A_{LF(q,\lceil \log_2 m \rceil)} \rceil + A_{CPA(r)} + \\ + 2 \cdot A_{CSA(3,\lceil \log_2 M \rceil + 1)} + \ + 2 \cdot A_{CPA(\lceil \log_2 M \rceil)} + \\ + \lceil \log_2 M \rceil \cdot A_{MUX(2\text{-}1)}, \end{aligned} \tag{19}$$

where $A_{LF(q,l)}$ is the area of the block of q-variable l logic functions, $A_{CPA(r)} = r \cdot A_{FA}$, is the area of the Ripple Carry Adder(RCA), where A_{FA} denotes the area of the FA, and

$$A_{CSA(p,a)} = (p-2) \cdot \lceil \log_2 M \rceil \cdot A_{FA} + \lceil \log_2 p \rceil \cdot A_{FA}, \tag{20}$$

the area of the Wallace tree with p operands of $\lceil \log_2 M \rceil$-bits. The CPAs of $\lceil \log_2 M \rceil$-bits are assumed in the Column Compression (CC) form as in [10] with the area of $k \cdot \lceil \log_2 k \rceil \cdot A_{FA}$ for k-bit length. $A_{MUX(1\text{-}2)}$ is the area of a 2-input multiplexer, and this area is here given by $A_{MUX(1\text{-}2)} = 3 \cdot A_{NAND2d1} + A_{INVd4}$. In the considered realization of the converter q-variable logic functions with q=5 are needed. Such a function is implemented using two multiplexed functions of 4-variables realized jointly as a block with a regular structure denoted as $LF^{(q)}$. The details of this design are given in [15]. It is assumed that the block of five 4-variable logic functions is realized using two multiplexed functions of 3 variables for each function with a common generator of implicants for all functions of the block with the suitable buffering. Using the data from [17] we receive $A_{LF^{(4)}} = 87.7$ GE and $A_{LF^{(5)}} = 178$ GE.

5.2. Delay evaluation

The converter delay can be evaluated as follows

$$t_{conv} = t_{LF^{(q)}} + \Theta(\lceil l/q \rceil) \cdot t_{FA} + t_{CPA(d)} + t_{LF^{(d+1)}} + 2t_{FA} + t_{CPA(a)} + t_{MUX2\text{-}1}. \tag{21}$$

The delays of logic blocks are $t_{LF^{(4)}} = 0.90\,ns$ and $t_{LF^{(5)}} = 1.25ns$. $\Theta(\cdot)$ denotes the number of levels in the n-operand CSA tree([16], p.102). For the first CPA in (21) in the RCA form, due to the fast carry propagation in the FA from STDL 130 ($t_{FA} = 0.54ns$, $t_{CICO} = 0.28ns$), we have

$$t_{CPA(k)} \cong \lceil k/2 \rceil \cdot t_{FA}. \tag{22}$$

The second adder in the CC form [10] has the delay

$$t_{CPA(k)} \cong (\lceil \log_2 k \rceil + 1) \cdot t_{FA}. \tag{23}$$

The MUX(2-1) delay is equal to $t_{MUX(2-1)} = 0.7 \cdot t_{FA}$, for the data from [17].

6. Example and comparison

We shall compare the hardware amount and delay of the new converter with that from [8], for the RNS base:

$$B = \{32,31,29,27,25,23,19,17\}.$$

The dynamic range for this base is 37.07 bits.

The hardware amount of the converter from [8] can be expressed as

$$\begin{aligned} A_{conv\text{-}p} = n \quad & A_{ROM(2^a x \lceil \log_2 M \rceil)} + A_{CSA(n,\lceil \log_2 M \rceil)} + A_{ROM(2^r x \lceil \log_2 M \rceil)} + \\ & + 2 \cdot A_{CSA(3,\lceil \log_2 M \rceil)} + 2 \cdot A_{CPA(\lceil \log_2 M \rceil)} + \lceil \log_2 M \rceil \cdot A_{MUX(2\text{-}1),} \end{aligned} \quad (24)$$

where r, given in Table 1 for $n \le 8$, is the maximum number of bits of S_H and C_H, which determines the size of the modulo M generator (MSB converter) [8].

Tab. 1. The size r of the MSB converter in relation to number of moduli n
Tab. 1. Rozmiar konwertera MSB, r dla zadanej liczby modułów, n bazy RNS

n	r
4	6
5,6	7
7,8	8

The ROM area is evaluated using the estimates from [8]. However, the maximum number bits per word (bpw), p_{MAX}, should be relatively small. Here it is assumed that $p_{MAX} \le 8$, that gives good estimates. The validity of such estimation can in principle be evaluated by using ROM areas expressed in μm^2 [17]. These data, however, are only available only for certain ROM sizes.

The delay of the converter from [8] can be written as

$$\begin{aligned} t_{convp} = & t_{ROM(2^{\lceil \log m \rceil} \times p_{\max})} + \Theta(n) \cdot t_{FA} + t_{ROM(2^r \times p_{\max})} + \\ & + 2 \cdot t_{CSA(3,\lceil \log M \rceil)} + t_{CPA(\lceil \log M \rceil} + t_{MUX(2-1)}. \end{aligned} \quad (25)$$

The ROM delays expressed in t_{FA} units are $t_{ROM1} \cong 4.2 t_{FA}$, $t_{ROM2} \cong 4.6 t_{FA}$, for 5-bit and 8-bit address, respectively. These delays are quotients of the high-density synchronous diffusion programmable ROM access times and the worst case FA delay from [17].

In Table 2 below the hardware amounts, delays and maximum pipelining rates for three types of converters are given: for the new converter with memoryless (NML)realization and ROM-based (New-ROM) realization and the converter from [8] (Piestrak-ROM). In addition to the above analysis, also an approximate analysis of the FPGA memoryless implementation of the new converter for the Xilinx Virtex-II Pro [19] has been carried out.

Tab. 2. Comparison of converters
Tab. 2. Porównanie konwerterów

	NML	New-ROM	Piestrak-ROM
HA [A_{FA}]	1971	1476	1824
Delay [t_{FA}]	21.3	25.1	22.5
Pipelining rate [t_{FA}]	1	4.6	4.6

It has been observed that about 520 slices are required for the assumed 8-moduli RNS base. Thus the converter requires less than half of the resources of the smallest device of this family (XC2VP2 comprises 1408 slices).

7. Conclusion

An effective CRT-based residue-to-binary converter for five bit moduli has been presented. The proposed technique of reduction of the sum of orthogonal projections to the range [0,2M) allows for memoryless implementation and pipelining on the FA level. A slight increase in the hardware amount (5%) for the 37-bit RNS number range has been observed with a 5% reduction of the delay. For the ROM-based realization for the above number range the use of the new technique allows a hardware reduction by about 20% but at the cost of an increased delay by 10-15%.

8. Literatura

[1] Szabo N.S., Tanaka R.J.: Residue Arithmetic and its Applications to Computer Technology, New York, McGraw-Hill, 1967.

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

[3] Chacraborti, N.B. Soundaranajan J.S. and Reddy A.L.N.: An implementation of mixed-radix conversion for residue number systems applications, IEEE Trans. on Comput., Vol. C-35, August 1986.

[4] Barsi F., Pinotti M.C.: Time-optimal mixed radix conversion for residue number applications, Computer J., Vol. 37, no.10, 1994, pp. 907-916.

[5] Henkelmann H., Drolshagen A., Bagherinia H., Ahrens H., Anheier W.: Automated implementation of RNS-to-binary converters, IEEE ISCAS Conference, Naval Postgraduate School, Monterrey, CA, June 3, 1998.

[6] Zhang C.N., Shirazi B., Yun D.Y.Y.: Parallel designs for chinese remainder conversion, Proc. Int. Conf. on Parallel Processing, August 17-21, 1987, pp.557-559.

[7] Elleithy K.M., Bayoumi M.A.: Fast and flexible architectures for RNS arithmetic decoding, IEEE Trans. on Circuits and Systems-II: Analog and Digital Signal Processing, vol. 39, No.4, April 1992, pp. 226-235.

[8] Piestrak S.J.: Design of high-speed residue-to-binary number system converter based on the Chinese Remainder Theorem, Proc. ICCD'94, Int. Conf. on Computer Design: VLSI in Computers and Processors, Cambridge, MA, Oct. 10-12,1994, pp. 508-511.

[9] Wang Y.: Residue-to-binary converters based on the new Chinese Remainder Theorems, IEEE Trans. Circuits Syst. -II Analog and Digital Signal Processing, Vol. 47, March 2000, pp.197-205.

[10] Wang Z., Jullien G.A., Miller W.C.: An improved residue-to-binary converter, IEEE Trans. Circuits Syst. -I :Fundamental Theory and Applications,Vol. 47, Sept. 2000, pp.1437-1440.

[11] Meehan S.J., O'Neil S.D., Vaccaro J.J.: An universal input and output RNS converter, IEEE Trans. Circuits Syst. Vol. CAS-37, June1990, pp.1158-1162.

[12] Cardarilli G.C., Re M., Lojacono R. : A systolic architecture for high-performance scaled residue to binary conversion, IEEE Trans. Circuits Syst. -I :Fundamental Theory and Applications, vol. 47, October 2000, pp.667-669.

[13] Huang C.H.: A fully parallel mixed-radix conversion algorithm for the residue number system, IEEE Trans. on Comput. vol. C-32, April 1983, pp.398-402.

[14] Burgess N.: Scaled and unscaled residue number system to binary conversion techniques using the core function, 1997 IEEE Symposium on Computer Arithmetic, pp. 250-257.

[15] Czyżak M.: High-speed binary-to-residue converter with the improved architecture, SPETO 2004, May, Niedzica, Poland.

[16] Hwang K.: Computer Arithmetic, Wiley, 1979.

[17] Samsung Electronics: Standard Cell Logic Library STDL 130, 2001.

[18] Czyżak, M. : High-speed residue-to-binary converter based on the Chinese Remainder Theorem, Proc. of the Sixth Int. Scientific Conf. Electronic Computers and Informatics ECI 2004, Sept. 22-24, 2004, Kosice-Herl'any, Slovakia, pp, 212-207.

[19] Xilinx: Virtex-II Pro Platform FPGA Handbook, 2002.

Artykuł recenzowany