Zenon ULMAN*, Maciej CZYŻAK**
*GDAŃSK UNIVERSITY OF TECHNOLOGY, FACULTY OF ELECTRICAL AND CONTROL ENGINEERING, DEPARTMENT OF CONTROL ENGINEERING
**GDAŃSK UNIVERSITY OF TECHNOLOGY, FACULTY OF ELECTRICAL AND CONTROL ENGINEERING, DEPARTMENT OF THEORETICAL ELECTROTECHNICS AND INFORMATICS

Effective residue-to-binary converter with the Chinese Remainder Theorem

Dr hab. inż. Zenon ULMAN

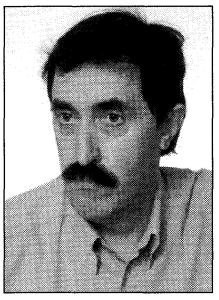
He received his Msc. degree from the Technical University of Gdansk (TUG) in 1970, and since 1972 he has been a researcher and lecturer at the Faculty of Electrical and Control Engineering. He received his Ph.D. in 1979 and the Doctor of Science degree in 2000 from the Technical University of Wroclaw. His scientific interests include computer arithmetic, number systems and fast digital signal processing.

ulman@ely.pg.gda.pl

Dr inż. Maciej CZYŻAK

He was graduated from the Technical University of Gdansk (TUG) in 1973, and received his Ph.D. in 1985. From 1984 to 1990 was a researcher and lecturer at the the Institute of Informatics, Faculty of Electronics. Since 1994 he has been a lecturer of computer science at the Faculty of Electrical and Control Engineering (TUG). His research interests encompass fast digital signal processing, computer arithmetic and VLSI design.

mczyzak@ely.pg.gda.pl



Abstract

The residue-to-binary conversion is the key operation in all digital signal processing applications that use the Residue Number System (RNS). In this work a new conversion technique based on the Chinese Remainder Theorem (CRT) for 5- and 6-bit moduli is proposed. It is especially suited for the realization with the use of binary arithmetic. The specific property of the technique is a way of calculation of the excess factor r in the CRT formula that makes possible, under certain conditions, the reduction of processed numbers from the range $[0, nM)$ to $[0,2M)$ where M is the product of moduli. This is done by replacing the calculation of r by the computation of the result of division of the sum of projections by a power of 2 close to M. Such approach allows for very effective hardware realization of the converter. Only small ROM's and standard binary adders are required. Moreover, the pipelining on the Full-Adder (FA) level is possible.

Streszczenie

Konwersja liczb z systemu resztowego do systemu binarnego jest podstawową operacją we wszystkich układach cyfrowego przetwarzania sygnałów, które wykorzystują system resztowy. W niniejszej pracy zaproponowano nową metodę konwersji opartą o chińskie twierdzenie o resztach dla modułów 5- i 6-bitowych. Specyficzną cechą nowej metody jest sposób obliczania tzw. współczynnika nadmiaru $r=\left[\sum_{j=1}^{n} X_j / M\right]$ w formule chińskiego twierdzenia o resztach, co umożliwia pod pewnymi warunkami, redukcję przetwarzanych liczb z zakresu $[0, nM)$ do $[0,2M)$, gdzie $M=\prod_{j=1}^{n} m_j$ Jest to realizowane poprzez zastąpienie obliczania r obliczaniem $r_B=\left[\sum_{j=1}^{n} X_j / M_B\right]$, gdzie M_B jest potęgą liczby 2 bliską M. Takie podejście pozwala na bardzo efektywną sprzętową realizację konwertera. Konieczne są tylko małe pamięci typu ROM i standardowe sumatory binarne. Ponadto możliwa jest realizacja potokowa z częstoliwością ograniczoną opóźnieniem sumatora 1-bitowego.

Keywords: digital signal processing, fast computer arithmetic, residue number system, residue-to-binary converter
Słowa kluczowe: cyfrowe przetwarzanie sygnałów, szybka arytmetyka komputerowa, resztowy system liczbowy

1. Introduction

There are many areas of science and technology in which add-multiply intensive, digital high-speed architectures are needed for processing of high-frequency signals in real-time. The exemplary applications are beamforming or high-resolution sampling of radio-frequency signals. The Residue Number System (RNS) [1] is advantageous for such applications since it permits high-speed low-level pipelined realization of addition, subtraction and multiplication. The RNS allows to replace operations in a large integer ring by the set of operations in smaller integer rings. Addition, subtraction are performed independently in the individual rings without carries between the positions of the number. In the RNS, the nonnegative integer X from the number range $[0, M-1]$ is represented by n-tuple $\left(|X|_{m_1}, |X|_{m_2}, \ldots, |X|_{m_n}\right)$, 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, $B=\{m_1, m_2, \ldots, m_n\}$ and $M=\prod_{j=1}^{n} m_j$. The moduli m_j should be pairwise coprime 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 CRT[1]. Each RNS-based architecture requires input and output processing, which usually ecompass the binary-to-residue and residue-to-binary conversions. The latter is much more complicated than the former. The conversions along with the other difficult operations in the RNS such as sign detection, magnitude comparison, overflow detection, scaling and division, have prevented the RNS use in general computing, limiting its application to the situations where it seems to be absolutely necessary. The residue-to-binary conversion has been implemented using the Mixed-Radix Conversion(MRC) algorithm [1, 2, 3, 4], or the CRT [1], [6-21], or the concurrent use of the CRT and MRS [5]. or the so-called core function [23]. The conversion based on the MRC is a relatively slow process in which first the RNS/MRC conversion is performed and subsequently the MRC/B. The CRT-based conversion is currently the most frequent approach. The main drawback is the necessity of performing the modulo M operation with M being a large number where the value to be reduced is the sum of orthogonal projections of the RNS number. In principle, there are two main approaches to the implementation. The first approach is to compute initially the sum of projections in the binary form and then perform a series of comparisons with $n-1$ multiplicities of M. Fraser and Bryg [6] limited the number of comparisons to one for a certain example and Cheng and Huang [7] proved for this technique that as the number of moduli increases, M should approach the power of 2. The second approach is to add every two operands in binary, detect overflow and make correction if necessary. For n operands the sum can be received using a tree of two-operand modulo M adders, but these devices are, in general, much more expensive than the binary adders. Taylor [8] assumed a special form of M, $M=2^k-2^l$, with k, l integers, that allowed for the realization of detection and correction with a simple logic but such a form of M severely limits the choice of the RNS base. Jenkins [9] proposed the computation of the sum with a bias in order to obtain the overflow in binary addition, indicated by the most significant carry bit, instead of overflow modulo M. Nevertheless the correction still may be needed. Zhang et al. [10] extended this method through the use of the redundant carry-save representations. Elleithy and Bayoumi [11] proposed an effective structure using a range determinator, so that the appriopriate value can be subtracted using the carry-look ahead adder. Piestrak [12] suggested an efficient reduction of the sum to the interval [0, 2M) using the carry-save representation and a modulo M generator for the number represented by the most significant bits of the carry and save vectors. This allows for the

final modulo M generation based on two binary adders working in parallel. This conversion technique uses only one $\lceil \log M \rceil$-bit addition in series (all logarithms here are logarithms to the base of 2). However, for the greater number of moduli this approach becomes less effective due to the growth of the ROM used in the structure. The new conversion technique was recently proposed by Y. Wang [13], but this approach does not seem to be effective since several multipliers are needed with the maximum size of $\lceil 0.5 \cdot \log M \rceil$-bits. In general, the complexity of the CRT computation depends on the number and the form of moduli. The DSP applications use two classes of moduli, the first are the moduli with the binary size $\lceil \log m \rceil \le 6$ bits, that can provide the sufficient parallelism and the processing structure can be based on small look-up tables and boolean logic only. The second group are moduli akin to the powers of 2, eg. $(2^k-1, 2^k, 2^k+1)$ or $(n-2^k, n+2^k)$. For the former group the most efficient converter was shown recently by Z.Wang et al. [14], whereas for the second group by Cardarilli et al.[15]. Such moduli simplify the conversion and allow also for the wider use of the binary arithmetic components, but the RNS parallelism is limited due to the small number of the RNS channels. Along with the CRT techniques based on the integer arithmetic, the techniques using the real arithmetic have been developed. First of them is the technique called the Approximate Chinese Remainder Theorem (ACRT) introduced by Soderstrand and Vernia [16], where the modulo M operation is avoided by using a sum of fractions that represent the orthogonal projections scaled by M. The conversion result is obtained as X/M. This technique was further developed and analyzed in [16, 17, 18, 19]. The other techique based on fractions, for conversion and sign detection was introduced by Vu [20]. A similar approach was taken in [21] for computation of the excess factor in the CRT formula, i.e. the number of times the number to be reduced overflows M. In this case the modulo M operation requires one binary adder operating in parallel with the calculation of the sum of projections, and a multiplier of a small number by M, implemented as a look-up table. Recently a technique based on the fractional approach was presented by Cardarilli et al. [22].

In this paper a new technique of computing X by the CRT is proposed. The technique makes use of the modulo M_B operation result with $M_B = 2^p$ instead of the modulo M operation. The concept can be used in the digital devices performing complex operations in residue arithmetic but with no limitation to this very area alone. In Section 2 the principle of the technique is shown, in Section 3 the computation in the binary arithmetic is derived and in Sections 4 and 5 the hardware realization is suggested along with the analysis of the hardware amount and processing delay.

2. New CRT computation

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

$$X = \left| \sum_{j=1}^{n} X_j \right|_M \tag{1}$$

or equivalently,

$$X = \left| \sum_{j=1}^{n} X_j \right|_M = \sum_{j=1}^{n} X_j - r \cdot M \tag{2}$$

where r is a nonnegative integer, $r < n$, such that

$$r = \frac{\sum_{j=1}^{n} X_j - \left| \sum_{j=1}^{n} X_j \right|_M}{M} = \left\lfloor \frac{\sum_{j=1}^{n} X_j}{M} \right\rfloor \tag{3}$$

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 $\gcd(M_j, m_j) = 1$.

It is seen from (1) and (2) that the value of X can be computed by the CRT in two, mathematically equivalent, ways. However, the realizations in arithmetic devices are different. The use of (1) requires the application of the multi-operand adder modulo M, where M is usually large. The digital realization of such an adder is complex and for this reason (1) is not taken here under consideration. The solution with the use of (2) which leads to a simple realization based on binary adders and small ROM's, is considered below.

Let X be an integer from the RNS number range [0,M-1], and let M_B be a power of 2 such that $M < M_B$ or $M > M_B$. Then we can write

$$\sum_{j=1}^{n} X_j = r_B \cdot M_B + X_B = r \cdot M + X \tag{4}$$

with $X = \left| \sum_{j=1}^{n} X_j \right|_M < M$, and $X_B = \left| \sum_{j=1}^{n} X_j \right|_{M_B} < M_B$.

Consider two cases.

CASE 1. $M < M_B$. It is intuitively evident, that in this case $r_B \le r$, but for completness it will be formally proven below.

LEMMA 1. If $M_B > M$ then $r_B \le r$.

Proof. We shall prove that (4) can not be fulfilled if it were $r_B > r$. Assume conversely that

$$r_B = r + \rho, \ \rho = 0, 1, 2, ..., \rho_{\max} \tag{5}$$

Inserting (5) into (4) we obtain the inequality $r \cdot M_B + \rho \cdot M_B + X_B > r \cdot M + X$, therefore (4) can not hold. Hence, $r_B \le r$.

Regarding LEMMA 1, we may write

$$r_B = r - \rho, \ \rho = 0, 1, 2, ..., \rho_{\max} \tag{6}$$

where ρ is related to $\partial = M_B - M$.

As $r < n$, hence in view of LEMMA 1, $r_B < n$. The lower bound of ρ is 0, and the upper bound is n-1, also $\rho \le r$.

Now we shall prove the theorem that shows how X can be computed in binary arithmetic using X_B, r_B, and ρ.

STATEMENT 1. The CRT given by (2) for $M_B > M$ is equivalent to the formula

$$X = X_B - \rho \cdot M_B + \partial \cdot r_B + \rho \cdot \partial = X_B + \delta \cdot r_B - \rho \cdot M \tag{7}$$

Proof. (7) can be received by inserting (6) into (4).

Because of hardware complexity when (7) is used, ρ should be as small as possible, but $\partial = M_B - M$ and the maximum value of ρ are related. In the following we shall determine the conditions under which $\rho \le 1$.

STATEMENT 2. Given M and M_B. If

$$(n-1) \cdot M > (n-2) \cdot M_B \tag{8}$$

then

$$\rho = r - r_B \le 1 \tag{9}$$

Proof. The maximum value of $\sum_{j=1}^{n} X_j$ can be expressed as

$$\max_{X \in [0, M-1]} \sum_{i=1}^{n} X_j = \max r \cdot M + X_{\max} = \max r_B \cdot M_B + X_{B\max} \tag{10}$$

where $X_{\max} = \left| \max \sum_{j=1}^{n} X_j \right|_M$, $X_{B\max} = \left| \max \sum_{j=1}^{n} X_j \right|_{M_B}$

In most cases that have the practical significance $\max \sum_{j=1}^{n} X_j$ belongs to the interval $[(n-1) \cdot M, n \cdot M]$ and it can be expressed as

$$\max \sum_{j=1}^{n} X_j = (n-1) \cdot M + X_{\max} \tag{11}$$

For the maximum value of ρ we have the following condition

$$\max \rho = \max r - \max r_B = \frac{\max \sum_{j=1}^{n} X_j - X_{\max}}{M} - \frac{\max \sum_{j=1}^{n} X_j - X_{B\max}}{M_B} \le 1 \tag{12}$$

$X_{B\max}$, if (8) holds, is expressed as

$$X_{B\max} = \begin{cases} (n-1) \cdot M + X_{\max} - (n-2) \cdot M_B, & \text{if } \sum_{j=1}^{n} X_j < (n-1) \cdot M_B \\ (n-1) \cdot M + X_{\max} - (n-1) \cdot M_B, & \text{if } \sum_{j=1}^{n} X_j > (n-1) \cdot M_B \end{cases} \tag{13}$$

Regarding (5), (8), (10),(13) we receive for both cases of $X_{B\max}$

$$(n-1)-\frac{(n-1)\cdot M+\max\sum_{j=1}^{n}X_j-X_{B\max}}{M_B}\le 1 \qquad (14)$$

CASE 2. $M_B < M$. In this case $r_B \ge r$. It is formally proven below.

LEMMA 2. If $M_B < M$ then $r_B \ge r$.

Proof. We shall prove that (4) can not be fulfilled if it were $r_B < r$. Assume conversely, that

$$r = r_B + \rho,\ \rho = 1,2,... \qquad (15)$$

Inserting(1) into (4) we obtain the inequality $r_B \cdot M_B + X_B < r_B \cdot M + \rho \cdot M + X$ and (4) can not hold. Hence, $r_B \ge r$.

Regarding LEMMA 2, we may write

$$r_B = r + \rho,\ \rho = 0,1,2,...,\rho_{\max} \qquad (16)$$

where $\rho_{\max}$,as in CASE 1, is related with $\partial' = M - M_B$.

As $r<n$, we have $r_B < n + \rho_{\max}$. Now we shall prove the theorem for the calculation of X using X_B, r_B, and ρ.

STATEMENT 3. The CRT given by (2), for $M_B < M$, is equivalent to the formula

$$X = X_B - r_B \cdot \delta' + \rho \cdot \delta' + \rho \cdot M_B = X_B - r_B \cdot \delta' + \rho \cdot M \qquad (17)$$

Proof. (17) can be obtained by insering (16) into (4).

We shall determine the conditions under which $\rho \le 1$.

STATEMENT 4. Given M and M_B, if

$$n \cdot M_B > (n-1) \cdot M \qquad (18)$$

then

$$\rho = r_B - r \le 1 \qquad (19)$$

Proof. Using (10) and (11) we may write (19) in the following form

$$\max\rho = \max r_B - \max r = \frac{\max\sum_{j=1}^{n}X_j - X_{B\max}}{M_B} - \frac{\max\sum_{j=1}^{n}X_j - X_{\max}}{M} \le 1 \qquad (20)$$

$X_{B\max}$, if (18) holds, is expressed as

$$X_{B\max} = \begin{cases} (n-1)\cdot M + X_{\max} - (n-1)\cdot M_B, & \text{if } \max\sum_{j=1}^{n}X_j < n\cdot M_B \\ (n-1)\cdot M + X_{\max} - n\cdot M_B, & \text{if } \max\sum_{j=1}^{n}X_j > n\cdot M_B \end{cases} \qquad (21)$$

Regarding (20), (10), (18), (21) we receive for both cases of $X_{B\max}$

$$\frac{(n-1)\cdot M+\max\sum_{j=1}^{n}X_j - X_{B\max}}{M_B} - (n-1) \le 1 \qquad (22)$$

3. Computation in binary arithmetic

Statements 1 and 2 can be applied to perform the calculation of X without the modulo M arithmetic and using binary arithmetic only. We remark that every X_j from (2) can be obtained in the binary form at the output of Look-Up-Tables (LUT's), which are addressed by the residues $|X|_{m_j}$. $\sum_{j=1}^{n}X_j$ can be received using an n-operand binary adder. The optimal structure of such an adder is the Carry-Save Adder (CSA) tree with the two-operand Carry-Propagate Adder(CPA) in the Carry-Look-Ahead (CLA) adder form as the final stage. At the output of the CLA, for $M < M_B$, we obtain

$$\sum_{j=1}^{n}X_j = X + r\cdot M = X_B + r_B \cdot M_B = X_B + r_B \cdot 2^p = \sum_{i=0}^{l-1} d_i \cdot 2^i < n\cdot M_B = n\cdot 2^p \qquad (23)$$

where $l = p + \lceil \log n \rceil$. X_B and r_B can be obtained as below

$$X_B = \left|\sum_{j=1}^{n}X_j\right|_{M_B} = \left|\sum_{i=0}^{l-1} d_i \cdot 2^i\right|_{2^p} = \left|X_B + r_B \cdot 2^p\right|_{2^p} = \sum_{i=0}^{p-1} d_i \cdot 2^i \qquad (24)$$

$$r_B = \frac{\sum_{j=1}^{n}X_j - X_B}{M_B} = \frac{\sum_{i=0}^{l-1} d_i\cdot 2^i - \sum_{i=0}^{p-1} d_i \cdot 2^i}{2^p} = \frac{\sum_{i=p}^{l-1} d_i \cdot 2^{i-p}\cdot 2^p}{2^p} = \sum_{i=p}^{l-1} d_i \cdot 2^{i-p} = \sum_{k=0}^{l-p-1} d_k \cdot 2^k \qquad (25)$$

where $k = i - p$.

From (23), (24) and (25) it is seen that the binary representations and r_B can be obtained from the binary representation of $\sum_{j=1}^{n}X_j$ as the first p least significant bits and $l-p$ most significant bits of X_B, respectively. Only binary adders and ROM's are needed.

The formulas for computation of X when the condition $M > M_B$ is fulfilled, can be derived similarly.

4. Hardware realization

Once X_B and r_B are given, for $\rho_{\max} = 1$, the final modulo generation of X can be based on STATEMENT 1 for $M < M_B$ if (8) is fulfilled. We get the following formulas

$$X = X_B + \delta \cdot r_B \text{ for } \rho = 0 \qquad (26a)$$

$$X = X_B + \delta \cdot r_B - M \text{ for } \rho = 1 \qquad (26b)$$

If (18) is fulfilled the generation can be based on STATEMENT 3 as follows

$$X = X_B - \delta \cdot r_B \text{ for } \rho = 0 \qquad (27a)$$

$$X = X_B - \delta \cdot r_B + M \text{ for } \rho = 1 \qquad (27b)$$

In each pair only one formula gives the correct value of X, and this value should be selected. It is assumed that both formulas are computed in parallel, and the nonnegative value is selected as the correct result. The converter structure is shown in Fig. 1.

The circuit operates as follows. The sum of the orthogonal projections, $\sum_{j=1}^{n}X_j$ is computed using Carry-Save-Adder tree with the subsequent two-operand Carry-Look-Ahead (CLA) adder. p least significant bits of the sum represent X_B, and $l-p$ most significant bits represent r_B. Next, r_B, represented by $[\log n]$-bits, is multiplied by ∂. Since $[\log n]$ usually does not exceed 4, the multiplication can be done by simple logic or a very small ROM. Two possible values of X are computed by two parallel CLA adders and then the correct value appears at the output of the multiplexer (MUX) controlled by the carry bit of the left-hand adder that computes $X_B + \partial \cdot r_B - M$.

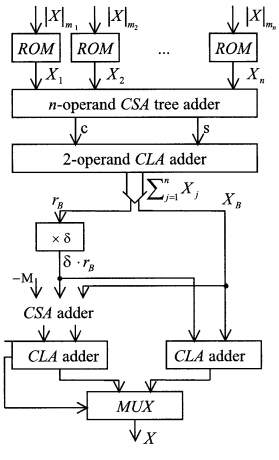


Fig. 1. The hardware realization of the CRT
Rys. 1. Sprzętowa realizacja CRT

5. Time-hardware complexity of new converter

The time-hardware complexity of a VLSI circuit can be considered in various ways in dependence on its form. For the prospective FPGA implementation the number of Configurable Logic Blocks (CLBs) [24] used as logic functions, FAs, registers or memory can be estimated along with other resources. Such analysis allows to select the appriopriate size of the FPGA. For ASICs based on Standard Cell Library(STCL), the area occupied by the logic elements, expressed in μm^2 or GE(Gate Equivalents), where the GE is the area of two-input NAND with fan-out=1. Equivalently, the GE can be expressed in transistors. The STCL allows also a rough estimate of power consumption for the required frequency of operation. Here the complexity analysis will be based on the ROM model used in [12], and logic components from Samsung 0.18μ STDL 130 [25].

A. ROM model

The approximate area of the $ROM(2^m \times p)$, $A_{ROM(2^m xp)}$ is expressed in transistors as

$$A_{ROM(2^m xp)}=([m/2]+1)\cdot 2^{[m/2]}+p\cdot 2^m+p\cdot([m/2]+2)\cdot 2^{[m/2]}+p\cdot(2^{[p/2]}+1) \quad (28)$$

The delay of a $ROM(2^m x\ p)$, using the *FA* delay, t_{FA} as the unit, can be expressed as

$$t_{ROM(2^m x\ p)}=(1+[\log_2 m]+[m/2])/2 \quad (29)$$

B. Models of logic components

In Table 1 the data of the individual logic components is shown which will be used in the analysis.

The symbol NANDxD1 denotes the x-input NAND gate with fan-out=1 (1 drive).

Table 1. The hardware amount in GE and maximum delays of the logic components Samsung 0.18μ STDL 130
Tabela 1. Ilość sprzętu w GE i maksymalne opóźnienia elementów logicznych z biblioteki Samsung 0.18μ STDL 130

Gate	NAND2D1	NAND3D1	NAND4d1	NAND8D1	INVD1	INVD4	FA
GE	1.00	1.33	1.67	4.33	1.00	1.67	7.33
Delay [ns]	0.13	0.18	0.23	0.31	0.12	0.08	0.54

C. The hardware complexity of the new converter

It is assumed that all moduli of the RNS base have the same binary length $b=\lceil \log m_j \rceil \le 5, j=1,2,...,n$.

The hardware amount of the new converter A_{conv} can be expressed in the following manner

$$A_{conv}=n\cdot A_{ROM(2^b x[\log nM])}+A_{CSA(n)}+A_{CPA([\log nM])}+A_{t_B\delta}+A_{CSA(1)}+ +2\cdot A_{CPA([\log M])}+[\log M]\cdot A_{MUX(2-1)} \quad (30)$$

where $A_{CSA(n)}=(n-2)\cdot[\log nM]\cdot A_{FA}+[\log n]\cdot A_{FA}$, and A_{FA} is the FA area, $A_{CPA(k)}=k\cdot[\log k]\cdot A_{FA}$, the CPA is assumed in the form as in [14], $A_{MUX}=3\cdot A_{NAND2D1}+A_{INVD1}$

The $r_B\delta$-block (Fig. 1) can be implemented as the $ROM(2^{[\log r_B]}x[\log r_{BMAX}\delta])$ or as the block of $[\log r_{BMAX}\delta]$ logic functions of $[\log r_{BMAX}]$ variables. For the *ROM*-based realization we obtain

$$A_{r_B\delta-ROM}=[[\log r_{BMAX}\delta]/p_{MAX}]\cdot A_{ROM(2^{[\log r_B]}xp_{MAX})} \quad (31)$$

where $p_{\max}$ is the maximum size of the ROM output word. Such approach seems to be more flexible since it is, in general, difficult to select the *ROM* of the appriopriate size in the given standard cell library. For the implementation based on logic functions we have

$$A_{r_B\delta-LF}=[\log r_{BMAX}\cdot\delta]\cdot A_{LF([\log r_{BMAX}])} \quad (32)$$

where $A_{LF([\log r_{BMAX}])}$ denotes the hardware amount of the $[\log r_{BMAX}]$-variable logic function. It is enough to consider only 3- and 4-variable logic functions as the maximum number of coprime moduli, with the binary length not exceeding 5 bits, is 10. For logic functions the general form is assumed in which all functions of the given number of variables can be realized. The 3-layer NAND realizations are assumed in the following form

3-variable logic function 3 INVD4- 4 NAND3D1- 1 AND4D1 (or 1 AND 4D1),

4-variable logic function 4 INVD4- 4 NAND4D1- 1 NAND8D1 (or 1 AND-8D1).

In the above description the number of gates is given for each layer. This approach is due to the fact that the upper bound is sought for and all logic functions of the given size can be realized by changing the connections of the input inverters and the gate in the third layer.

D. The delay of the new converter

The delay of the new converter can be expressed as follows

$$t_{conv}=t_{ROM(2^{[\log m]}xp_{MAX})}+t_{CSA(n)}+t_{CPA([\log nM])}+t_{r\delta}+ +t_{CSA(1)}+t_{CPA([\log nM])}+t_{MUX(2-1)} \quad (33)$$

where

- $t_{ROM(2^{[\log n]}xp_{MAX})}$
- $t_{CSA(n)}$ is $\theta(n)t_{FA}$, where $\theta(n)$ is the number of layers in the n-operand.

CSA tree computed using the recursive formula from [11], shown in Table 2.

Table 2. The maximum numbers of operands, n_{MAX} for the number of layers $\theta(n)$ in the CSA tree for $n=1, 2, ..., 8$
Tabela 2. Maksymalna liczba operandów, n_{MAX} dla liczby warstw $\theta(n)$ w drzewie CSA dla $n=1,2,...,8$

$\theta(n)$	1	2	3	4	5	6	7	8
n_{MAX}	3	4	6	9	13	19	28	42

- $t_{CPA([\log nM])}=(1+\log\log nM)\cdot t_{FA}$
- $t_{r\delta}=t_{ROM(2^{[\log n]}xp_{MAX})}$ or $\approx t_{FA}$ for the realization based on logic functions
- $t_{MUX(2-1)}=t_{INVD1}+2\cdot t_{NAND2D1}$

6. Numerical examples and comparison

We shall compare here the hardware amount and delay of the new converter with the Piestrak converter from [12], for the three RNS bases: $B_1=\{32,31,29,27,19\}$, $B_2=\{32,31,29,27,25,23,19\}$ and $B_3=\{32,31,29,27,25,19,17,13,11,7\}$. The dynamic ranges for these bases are approximately 24, 33 and 47 bits, respectively. We have to select for each base appriopriate M_B, so that the condition (8) or (18) is fulfilled, if it is possible at all for the given base. For the first base we have $M_{B1}=\prod_{j=1}^{5}m_j=14757984$, and $2^{24}=16777216$, and (8) is fulfilled as $4\cdot 14757984>3\cdot 16777216$; similarly, for the second base we obtain $M_{B2}=\prod_{j=1}^{7}m_j=8485840800$ and $2^{33}=8589934592$, with $6\cdot 8485840800>5\cdot 8589934592$, whereas for the third base we have $M_{B3}=\prod_{j=1}^{11}m_j=1444035528993600$ and, we see that M_{B3} is close to 2^{47} but $M_{B3}>2^{47}$, and so as (8) is not fulfilled, we have to examine if the relation for the CASE 2 can be used, i.e. (18). We find that $11\cdot 140737488355328>10\cdot 1444035528993600$.

The hardware amount of the converter from [12] can be expressed in the following form

$$A_{conv\,p}=n\cdot A_{ROM(2^b x[\log nM])}+A_{CSA(n)}+A_{ROM(2^r x[\log M])}+2A_{CSA(1)}+ +2\cdot A_{CPA[\log M]}+[\log M]\cdot A_{MUX(2-1)}$$

where r is the size in bits of the MSB converter; for $n_1=5$, $n_2=7$, and $n_3=11$, we have $r_1=7$, $r_2=8$, and $r_3=9$, and the delay

$$t_{conv}=t_{ROM(2^{[\log m]}xp_{MAX})}+t_{CSA(n)}+t_{ROM(2^r xp_{MAX})}+2t_{CSA(1)}+ +t_{CPA([\log nM])}+t_{MUX(2-1)}$$

Using these relationships for the converter from [12] and models from Section V for the new converter the hardware amounts, delays and maximum pipelining rate for both types of the converter and the individual bases have been calculated. The obtained hardware amounts expressed in transistors are shown in Table 3, and the delays in Table 4.

Table 3. Hardware amounts in transistors.
Tabela 3. Ilość sprzętu wyrażona w tranzystorach.

RNS base	B_1	B_2	B_3
New converter	22179	39345	75817
Converter from [12]	22674	45692	99621

It is seen that the new converter has the lower hardware complexity for the larger dynamic ranges, but it is about 10% slower for all considered RNS bases. It was assumed that $r\delta$-block is implemented using 3-variable logic functions for B_1, and 4-variable for B_3. The maximum pipelining rates are presented in Table 5. The unit for the numerical values is the FA delay, t_{FA}.

Table 4. Delay in t_{FA}
Tabela 4. Opóźnienie w t_{FA}

RNS base	B_1	B_2	B_3
New converter	21.5	24.5	25.5
Converter from [12]	19	21.5	24

Table 5. Maximum pipelining rates
Tabela 5. Minimalne okresy próbkowania

RNS base	B_1	B_2	B_3
New converter	$3.5+t_L$ $(1+t_L)$	$3.5+t_L$ $(1+t_L)$	$3.5+t_L$ $(1+t_L)$
Converter from [12]	$3.5+t_L$	$4+t_L$	$5.5+t_L$

t_L - is the latch delay. For the new converter the values in the parentheses denote the pipelining rates for the memoryless converter, i.e., such in which it was assumed that the input look-up tables are implemented using the 5-variable logic functions. However, they have to be individually designed since the use of the models similar to those presented in this work would increase the hardware amount by 90-100%.

7. Conclusions

An effective CRT-based residue-to-binary conversion technique for five- and six-bit moduli, but not limited to this class, is proposed. The effectiveness is due to the new algorithm for the reduction of the sum of orthogonal projections to the range $[0,2M)$. The algorithm is based on the indirect calculation of the excess factor in the CRT formula with the use of the directly given quotient of the sum of projections by the integer equal to a power of 2. Fast small ROM's and fast parallel binary adders are used in the hardware realization only. The maximum ROM address size does not exceed the binary size of the moduli. The proposed converter architecture has the lower asymptotic hardware complexity than the other known architectures. The time delay for the smaller number ranges is by 10-15% greater than that for the fastest architectures. The maximum pipelining rate is higher than for the other high-speed realizations. In the extreme case, the FA level pipelining can be attained when the input look-up tables are implemented using boolean circuits, however in this case the logic minimization is required.

References

[1] S.Szabo and R.J.Tanaka: Residue Arithmetic and its Applications to Computer Technology, New York, McGraw-Hill, 1967.
[2] N.B. Chacraborti, J.S. Soundaranajan and A.L.N. Reddy: An implementation of mixed-radix conversion for residue number systems applications, IEEE Trans. on Comput., vol. C-35, August 1986.
[3] F.Barsi, M.C. Pinotti: Time-optimal mixed radix conversion for residue number applications, Computer J., vol. 37, no.10, 1994, pp. 907-916.
[4] H.Henkelmann, A.Drolshagen, H.Bagherinia, H.Ahrens, W.Anheier: Automated implementation of RNS-to-binary converters, IEEE ISCAS Conference, Naval Postgraduate School, Monterrey, CA, June 3, 1998.
[5] C.H. Huang: A fully parallel mixed-radix conversion algorithm for the residue number system, IEEE Trans. on Comput. vol.C-32, April 1983, pp.398-402.
[6] D.F.Fraser, N.J.Bryg: An adaptive digital signal processor based on the residue number system, Proc. AIAA 2nd Comput. Aereospace Conf., Los Angeles, CA, Oct. 22-24,1979.
[7] V.S. Cheng, C.H. Huang,: On the decoding of residue number, Proc. Int. Symp. Mini- Microcomput. Contr. Measurement, San Francisco, CA, May 20-22, 1981.
[8] F.J. Taylor and A.S. Ramnarayanan: An efficient residue-to-decimal converter, IEEE Trans. Circuits Syst. vol. CAS-28, Dec. 1981, pp.1164-1169.
[9] W.K. Jenkins: A technique for the efficient generation of projections for error correcting residue codes, IEEE Trans. Circuits Syst. vol.CAS-30, pp. 223-226, Feb. 1984.
[10] C.N. Zhang, B. Shirazi, D.Y.Y. Yun: Parallel designs for chinese remainder conversion," Proc. Int. Conf. on Parallel Processing, August 17-21, 1987, pp.557-559.
[11] K.M. Elleithy, M.A. Bayoumi: Fast and flexible architectures for RNS arithmetic decoding, IEEE Trans. on Cicuits and Systems-II: Analog and digital signal processing, vol. 39, No.4, April 1992, pp. 226-235.
[12] S.J. Piestrak: 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.
[13] Y. Wang: 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.
[14] Z. Wang, G.A.Jullien, W.C.Miller: An improved residue-to-binary converter, IEEE Trans. Circuits Syst. -I:Fundamental Theory and Applications,vol. 47, Sept. 2000, pp.1437-1440.
[15] G.C. Cardarilli, M.Re, R.Lojacono: RNS-to binary conversion for efficient VLSI implementation, IEEE Trans. Circuits Syst. -I:Fundamental Theory and Applications, vol. 45, June 1998, pp.667-669.
[16] M.A. Soderstrand, C.Vernia and J. –H. Chang: An improved residue number system digital-to-analog converter: IEEE Trans. Circuits Syst. vol. CAS-30, 1983, pp.903-907.
[17] S.J. Meehan, S.D. O'Neil, J.J. Vaccaro: An universal input and output RNS converter, IEEE Trans. Circuits Syst. vol. CAS-37, June1990, pp.1158-1162.
[18] J.Y. Kim, K.H.Park, H.S.Lee:Efficient residue-to-binary conversion technique with rounding error compensation, IEEE Trans. Circuits Syst. vol. CAS-38, March 1991, pp.315-317.
[19] I.Lee, W.K. Jenkins:The design of residue number system arithmetic units for VLSI adaptive equalizer, Proc. of the Great Lakes Symposium on VLSI, Feb. 1998, pp.179-184.
[20] T.V. Vu: Efficient implementations of the Chinese Remainder Theorem for sign detection and residue decoding, IEEE Trans. Comput, vol.C-34, pp. July 1985, pp. 646-651.
[21] Z.D. Ulman and M.Czyżak: Highly parallel, fast scaling of numbers in nonredundant residue arithmetic, IEEE Trans. Signal Processing, vol. 46, pp.487-496, Feb. 1998.
[22] G.C. Cardarilli, M.Re, R.Lojacono: 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.
[23] N. Burgess: Scaled and unscaled residue number system to binary conversion techniques using the core function, 1997 IEEE Symposium on Computer Arithmetic, pp. 250-257.
[24] Xilinx Inc.: Virtex II Pro FPGA, 2003.
[25] Samsung Electronics: 0.18 Standard Cell Logic Library STDL 150, 2001.

Tytuł: Efektywna konwersja liczb z systemu resztowego do systemu wagowego z użyciem chińskiego twierdzenia o resztach

Artykuł recenzowany