PRACE NAUKOWE Akademii im. Jana Długosza w Częstochowie
Technika, Informatyka, Inżynieria Bezpieczeństwa 2017, t. V, s. 155–163

http://dx.doi.org/10.16926/tiib.2017.05.13

Mikhail Selianinau
Wydział Matematyczno-Przyrodniczy
Akademia im. Jana Długosza
al. Armii Krajowej 13/15, 42-200 Częstochowa
e-mail: m.selianinov@ajd.czest.pl

THE BASIC STRUCTURAL ELEMENTS OF MODULAR DEVICES FOR DIGITAL INFORMATION PROCESSING

Abstract. In this paper, we consider four basic methods for design of the modular adders, subtractors and multipliers suitable for modular number system: the direct logical method, the adder method, the ring shift method and the tabular method. It is shown that the variant of look-up table implementation of modular arithmetic operations using permanent storage devices is the simplest and most effective for organization of high-speed pipeline digital information processing.
Keywords: Modular arithmetic, modular number system, modular computing structures, modular operations, look-up table data processing.

BAZOWE ELEMENTY STRUKTURALNE URZĄDZEŃ MODULARNYCH DO CYFROWEGO PRZETWARZANIA INFORMACJI

Streszczenie. W niniejszym artykule przedstawiono cztery główne sposoby projektowania modularnych sumatorów, subtraktorów oraz mnożników: bezpośrednia metoda logiczna, metoda oparta na podstawie sumatorów binarnych, metoda przesunięcia cyklicznego oraz metoda tabelaryczna. Wykazano, że wariant tabelarycznej realizacji operacji arytmetycznych z zastosowaniem pamięci tylko do odczytu jest najbardziej prostym i skutecznym rozwiązaniem w zakresie szybkiego potokowego przetwarzania informacji cyfrowej w resztowych systemach liczbowych.
Słowa kluczowe: arytmetyka modularna, modularne systemy liczbowe, modularne struktury obliczeniowe, operacje modularne, tabelaryczne przetwarzanie informacji.

Introduction

The performance increase and the functionality expansion have always been and now are the main goals of the developing process of computing machinery. The implementation of new and more efficient methods of organization and realization of computation processes is one of the main ways to achieve these goals along with improvement in computer engineering. The analysis of the known approaches used in the development of high-performance computing structures shows that all of them have one common distinctive feature, which essence consists in the application of certain forms of parallel digital information processing.

Among all numerical systems, the number systems with a parallel structure are the most suitable for carrying out the parallel computations, especially the high-speed ones. First of all, the modular number systems (MNS), which are characterized by the maximum level of internal parallelism, belong to such systems [1−5]. The unique property of MNS to perform a natural decomposition of computational processes into independent components of less complexity led to the widespread use of modular arithmetic (MA) in modern computer science and its applications as an effective mathematical apparatus for a mapping of computational processes into the high-speed parallel architectures.

The basic notation and terminology

Let us introduce the following notation:

$\mathbf{Z}$ is the set of all integers;

$\lceil x \rceil$ denotes the ceiling of x, i.e. the smallest integer greater than or equal to x, $\lceil x \rceil = \min \{y \in \mathbf{Z} \mid y \geq x\}$;

$\mathbf{Z}_m = \{0, 1, \ldots, m-1\}$ is the set (ring) of least nonnegative residue modulo $m > 1$;

$\mathbf{Z}_m^* = \{1, 2, \ldots m-1\}$ is the cyclic group of the ring $\mathbf{Z}_m$;

$\varphi(m)$ is the Euler function (the number of positive residues of the ring $\mathbf{Z}_m$ relatively prime to m);

$|x|_m$ denotes the element of $\mathbf{Z}_m$ congruent to x modulo m;

$m_1, m_2, \ldots, m_k$ are the natural modules ($k \geq 2$);

The modern approaches for implementation the modulo operations

One of the most important characteristics of MA caused by the parallelism of the MNS structure is that the process of performing any operation on

numbers in a modular code, both modular and non-modular, reduces to a sequence of sets of single-step operations of the form

$$\gamma = F(\alpha, \beta), \tag{1}$$

where F is some function; α, β and γ are the nonnegative integer variables whose maximum values are commensurable with the values of the used modules of MNS. Various special methods which ensure high speed can be applied to implement the operations described by the relation (1) due to the smallness of α and β.

The adders, subtractors and multipliers with respect to MNS modules are the most common functional units (FU) of modular computing devices. There are four main methods for designing modular units: the direct logical method, the adder method, the ring shift method as well as the table method [3].

The direct logical method

In the direct logical design method, the modular operations are described at a level of switching function systems by means of which the binary digit values of the sums, differences and products of residues with respect to the corresponding modules are formed.

The result γ of the realized operation of the type $\gamma = O(\alpha, \beta)$ ($\alpha \in \mathbf{Z}_{m_i}$, $\beta \in \mathbf{Z}_{m_i}$; $i, j \in \mathbf{Z}_k$; O is a known function) is obtained in the binary code $\gamma = (\gamma_{\lambda-1}\, \gamma_{\lambda-2}\, \dots\, \gamma_0)_2$ by using the previously synthesized Boolean relations

$$\gamma_i = O_l\left(\alpha_0^{(i)}, \alpha_1^{(i)}, \dots, \alpha_{\lambda_i-1}^{(i)}; \beta_0^{(i)}, \beta_1^{(i)}, \dots, \beta_{\lambda_j-1}^{(i)}\right) (l = 0, 1, \dots, \lambda - 1), \tag{2}$$

where O_l is some switching function; $\alpha_0^{(i)}, \alpha_1^{(i)}, \dots, \alpha_{\lambda_i-1}^{(i)}$ and $\beta_0^{(i)}, \beta_1^{(i)}, \dots, \beta_{\lambda_j-1}^{(i)}$ are the binary digits of the operands α and β, respectively; $\lambda_i = \lceil log_2 m_i \rceil$, $\lambda_j = \lceil \log_2 m_j \rceil$, λ is the code length of the value γ.

Modular adders, subtractors and multipliers are implemented in the form of combinational circuits, which are synthesized within the traditional approach applied in the case of binary digital devices. As a rule, the expressions (2) have a simple form and allow an effective realization by means of the simple programmable logical arrays (PLA) due to the small values of the parameters λ_i and λ_j.The modular matrix structures on this basis are distinguished by simplicity, regularity and high competitiveness. The pipeline information processing is easily organized within these structures.

The adder method

This method implies that the binary adders are provided with the additional logic circuits that enable the addition of the residues with respect to the selected modules, at the same time all the arithmetic operations are carried out during a certain number of additions. The adder method is realized most simply with the use of modules of the form $m = 2^b, 2^b \pm 1$ (b is a natural number) and its application is reasonable only for sufficiently large m.

The idea underlying the adder method of the organization of computation processes in modular blocks consists in the use of the index principle of performance of the modular calculations and the binary adders as the basic FU. The isomorphism of the multiplicative group $\mathbf{Z}_m^*$ and the additive group $\mathbf{Z}_{m-1}$ set by the mapping $G : \mathbf{Z}_m^* \to \mathbf{Z}_{m-1}$, which assigns to each element $\chi \in \mathbf{Z}_m^*$ the unique element $G(\chi) = \mathrm{ind}_m \chi$ from $\mathbf{Z}_{m-1}$ satisfying the equality

$$\left| g^{\mathrm{ind}_g \chi} \right|_m = \chi, \tag{3}$$

plays a significant part in practical realization of the mentioned idea [6]. Here, g is a primitive root modulo m defined as an element of commutative group $\langle \mathbf{Z}_m^*, \times \rangle \subset \langle \mathbf{Z}_m, + \rangle$ with the order $N = \varphi(p) = m - 1$; the element $\mathrm{ind}_g \chi$ of the group $\mathbf{Z}_{m-1}$ is called an index or discrete logarithm of the number χ to the base g modulo m. According to (3), in the case of prime m for any $\alpha, \beta \in \mathbf{Z}_m^*$ the following relation is true

$$|\alpha\beta|_m = \left| g^{\mathrm{ind}_g |\alpha\beta|_m} \right|_m = \left| g^{\left| \mathrm{ind}_g \alpha + \mathrm{ind}_g \beta \right|_{m-1}} \right|_m. \tag{4}$$

Thus, the multiplication modulo prime m can always be reduced to modular addition because of the isomorphism of multiplicative group $\mathbf{Z}_m^*$ and additive index group $\mathbf{Z}_{m-1}$. According to (4), in order to obtain the result of the modular multiplication $|\alpha\beta|_m$ it is enough to determine the indices $\mathrm{ind}_g \alpha$ and $\mathrm{ind}_g \beta$ of the operands α and β, add them modulo $m - 1$ and then transform the resulting residue $\mathrm{ind}_g |\alpha\beta|_m = \left| \mathrm{ind}_g \alpha + \mathrm{ind}_g \beta \right|_{m-1}$ into the desired product $|\alpha\beta|_m$. For small value of the module m, the direct and inverse transformations of the residues χ and $\mathrm{ind}_g \chi$ ($\chi \in \mathbf{Z}_m$) corresponding to the mappings
$G: \chi \to \mathrm{ind}_g \chi$ and $G^{-1}: \mathrm{ind}_g \chi \to \chi$ in practice are easily carried out by the look-up table method [2, 3]. Since the volume of the tables required for transformations is about 10 times less than it is for the table of residue multiplication modulo m, then a considered method of modular multiplication with the use of indices

appears more efficient than a direct table method, especially with increase of the value of module m.

The advantages of the adder method become even more appreciable if the addition of the indices $\mathrm{ind}_g\,\alpha$ and $\mathrm{ind}_g\,\beta$ is performed not modulo m–1 but with respect to the set of pairwise prime modules whose product coincides with m–1. We note that in view of the simplicity of m, the module $m-1$ is composite. The use of specified decomposition procedure for the formation of $\mathrm{ind}_g|\alpha\beta|_m$ allows us to optimize the corresponding adder architectures both in complexity and speed.

The ring shift method

Another approach to creation of modular blocks, at which the data processing is carried out exclusively in a unitary code using only registers, is based on the use of certain properties of the residue ring $\mathbf{Z}_m$ as well as the tables of addition, subtraction and multiplication operations modulo m, i.e. the matrices $[|\alpha+\beta|_m]$, $[|\alpha-\beta|_m]$, $[|\alpha\times\beta|_m]$ $(\alpha,\beta\in\mathbf{Z}_m, m>1)$. An example of a matrix structure of addition and subtraction operations for the module $m=5$ is shown below:

$$[|\alpha+\beta|_m]=\begin{bmatrix}0&1&2&3&4\\1&2&3&4&0\\2&3&4&0&1\\3&4&0&1&2\\4&0&1&2&3\end{bmatrix},[|\alpha-\beta|_m]=\begin{bmatrix}0&4&3&2&1\\1&0&4&3&2\\2&1&0&4&3\\3&2&1&0&4\\4&3&2&1&0\end{bmatrix}. \quad (5)$$

As it may be seen from (5), the αth matrix row of the modular addition (subtraction) can be obtained by the β-fold cyclic shift to the left (right) of the elements of the zero row.

Let us consider the algorithm of formation of the modular sum $|\alpha+\beta|_m$ in a unitary code, which is determined as an m-bit code of the value $\chi\in\mathbf{Z}_m$ such that its χth digit is equal to 1.

The ring shift method is based on the isomorphism of the finite Abelian groups $\langle\mathbf{Z}_m;\,+\rangle$ and $\langle\mathbf{E}_m;\circ\rangle$ of the order m, where $\mathbf{E}_m=\{2^\chi\mid\chi\in\mathbf{Z}_m\}$ and the operation "$\circ$" is performed by the rule

$$a\circ b=2^{|\log_2 a+\log_2 b|_m}\ (a,b\in\mathbf{E}_m). \quad (6)$$

The required isomorphic mapping $U:\mathbf{Z}_m\to\mathbf{E}_m$ associates each residue $\chi\in\mathbf{Z}_m$ with an element $x=2^\chi$ of the set $\mathbf{E}_m$. At the same time, according to (6) for $\alpha,\beta\in\mathbf{Z}_m$ and $a,b\in\mathbf{E}_m$ the following equalities hold

$$U(\alpha) \circ U(\beta) = U(|\alpha + \beta|_m), \tag{7}$$

$$|U^{-1}(a) + U^{-1}(b)|_m = U^{-1}(a \circ b). \tag{8}$$

It follows from (7) and (8) that in order to obtain the modular sum $\gamma = |\alpha + \beta|_m$ it is sufficient to convert the operand α into the m-bit unitary code (i.e. the binary code of the number $U(\alpha)$) using the decoder. Further, the generated code is cyclically shifted by β bits to the left (towards the higher bits) in the m-bit ring shift register, and then the resultant unitary code (i.e. the binary code of $U(\gamma)$) is transformed into the binary code of the required residue γ using the encoder.

Using the ring shift method it is possible to perform not only addition but also subtraction of the elements of the set $\mathbf{Z}_m$. In contrast to the sum $|\alpha + \beta|_m$, when obtaining the modular difference $|\alpha - \beta|_m$ of residues α and β the m-bit unitary code $U(\alpha)$ is cyclically shifted by β bits not to the left but to the right, i.e. in the direction of the lower bits.

The realizations of this type have a regular structure and are well-suited for chips implementation.

Let us notice that in the case of the modular sum

$$R = \left| \sum_{l=1}^{\nu} R_l \right|_m \quad (R_l \in \mathbf{Z}_m), \tag{9}$$

consisting of $\nu > 2$ terms the effectiveness of the application of the ring shift method substantially increases due to the simplicity of accumulating the results of the summations performed during the calculation (9). This is clearly illustrated by the following implementation algorithm of the expression (9):

1) convert R_1 into the unitary code ($R_1 \to U(R_1)$) and assign the initial value $U(R_1)$ to the m-bit variable r ($r = U(R_1)$);

2) assign the value 2 to the index l ($l = 2$);

3) shift cyclically the binary code of r by R_l bits to the left;

4) if the equality $l = \nu$ holds, then go to the step 5); otherwise increment the index l by 1 ($l = l + 1$) and go to the step 3);

5) convert the unitary code of r into the binary code ($r \to U^{-1}(r)$) and assign the sought-for value $U^{-1}(r)$ to the variable R ($R = U^{-1}(r)$).

If the actions provided in items 1) and 2) of the given algorithm are replaced by the following sequence of operations:

- assign the initial state $r = U(0) = 2^0 = 1$ to the the m-bit variable r;
- assign the value 1 to the index l ($l = 1$),

then the calculation process of the expression (9) is carried out using only the cyclic shift operations and the encryption operation in the final step (see item 5) above).

As for the multiplication operation modulo m, the structure of the matrix $[|\alpha \times \beta|_m]$ is visible from the following example for $m = 5$:

$$[|\alpha \times \beta|_m] = \begin{bmatrix} 0\,0\,0\,0\,0 \\ 0\,1\,2\,3\,4 \\ 0\,2\,4\,1\,3 \\ 0\,3\,1\,4\,2 \\ 0\,4\,3\,2\,1 \end{bmatrix}. \qquad (10)$$

This operation can be reduced to modular addition because of the isomorphism of the multiplicative group $\mathbf{Z}_m^*$ and the additive index group $\mathbf{Z}_{m-1}$. Taking the above into account, it follows from (4) that in order to obtain the result $|\alpha\beta|_m$ of the modular multiplication in the case of prime m it is enough to generate the $(m-1)$-bit unitary code of the index $\mathrm{ind}_g\,\alpha$ with the help of decoder, cyclically shift it by $\mathrm{ind}_g\,\beta$ bits to the left and then transform the $(m-1)$-bit unitary code of the residue $\left|\mathrm{ind}_g\,\alpha + \mathrm{ind}_g\,\beta\right|_{m-1} = \mathrm{ind}_g|\alpha\beta|_m$ into the binary code of the desired modular product $|\alpha\beta|_m$ using an encoder. The defined approach can be generalized to composite modules.

It should be noted that the ring shift method can naturally be used to add the indices in the adder method considered above. In this case, the transformations $\alpha \to \mathrm{ind}_g\,\alpha$ and $\mathrm{ind}_g|\alpha\beta|_m \to |\alpha\beta|_m$ are changed by the transformations $\alpha \to 2^{\mathrm{ind}_g\,\alpha}$ and $2^{\mathrm{ind}_g|\alpha\beta|_m} \to |\alpha\beta|_m$, which are realized by a decoder and an encoder, respectively. If the concerned shift ring process (after obtaining $|\alpha\beta|_m$ in the $(m-1)$-bit shift register) is carried out with its initial content $U(0) = 1$, then only the second substitution of the transformations mentioned above is needed.

The table implementation using memory blocks

The considered designing methods of the modulo adders, subtractors and multipliers allow us to construct the modular blocks with high speed. However, the practical application of this blocks is expedient only when they are performed in the form of VLSI chips.

In the absence of the required VLSI chips, the tabular implementation of arithmetic operations in MNS using persistent storage devices (e.g., read only memory (ROM)) and other high-density storage devices is the simplest and most effective technique [2, 3]. In this way not only the modular operations of addition, subtraction and multiplication can be implemented but also

more complex operations, for example, the expressions of the form: $|F_1(\alpha) \pm F_2(\beta)|_m$ and $|F_1(\alpha) \times F_2(\beta)|_m$, where $F_1(\alpha)$ and $F_2(\beta)$ are some functions; α and β are the residues with respect to some modules. At the same time, for the most of the required table schemes the hardware costs can be substantially reduced due to the symmetry of modular operations.

If, for example, a certain ROM has a capacity of 2^b words of length b' bits, then it can be used to implement any function of the form $\chi' = F(\chi_1, \chi_2, \dots, \chi_l)$, where $\chi' \in \{0, 1, \dots, 2^{b'} - 1\}$; $\chi_i \in \{0, 1, \dots, 2^{b_i} - 1\}$; $(i = 1, 2, \dots, l)$; $\sum_{i=1}^{l} b_i \leq b$; $b, b', b_1, b_2, \dots, b_l$ and l are natural numbers. To do this, it is enough to write to the ROM memory the value $F(\chi_1, \chi_2, \dots, \chi_l)$ at the address $X = \chi_1 + \chi_2 2^{b_1} + \cdots + \chi_l 2^{b_1+b_2+\cdots+b_{l-1}}$ for all admissible values of the variables $\chi_1, \chi_2, \dots, \chi_l$. Then, when a set of values $\langle \chi_1, \chi_2, \dots, \chi_l \rangle$ is fed to the address inputs of the ROM, the required result χ' will be formed at its output.

In the case of pipeline information processing, the registers are connected to the outputs of the ROM. At the same time, the modular clock time t_{MT} is determined by the time t_{ROM} needed to read the word from the memory and the time t_R needed to store it in the register, i.e. $t_{MT} = t_{ROM} + t_R$.

Within the framework of the considered variant of tabular implementations, the ROMs in essence are unique logical elements and form the elemental basis of modular digital devices in combination with conventional logic elements as well as typical components of computing machinery which carry out digital information processing at the level of elementary operations on words. The modular computing structures based on the ROMs are distinguished by the uniformity, high degree of integration, simplicity of pipeline configurations; they provide a high speed and reliability and their effectiveness is steadily increasing with the improvement of integrated circuit technology.

Conclusions

The foregoing allows us to conclude that because of the isomorphism of the multiplicative group of the ring $\mathbf{Z}_m$ and the additive index group $\mathbf{Z}_{m-1}$ there is a principal possibility of constructing the modular blocks based only on the FU of practically the same type, for example, mainly on the basis of the ROM, PLM, shift registers or binary adders. This circumstance ensures exceptionally high manufacturability of modular computing architectures.

From a practical point of view, the fact that the resulting computing structures require only the use of FU operating with low-bit input values, when implementing the algorithms of MA, is of special interest. Since the execution of any operation on arbitrary set of residues with a relatively small total digit capacity can, in principle, always be done by the tabular way using the same type

of FU (ROM, PLM, etc.), then the hardware modular computing structures are characterized by an extremely high degree of uniformity and regularity, ideally consistent with the design features of the modern and prospective VLSI technologies, are well-suited to restructuring and organization of adaptive operation modes, have a low design cost as well as a number of other important advantages.

References

[1] Akushsky I.J., Yuditsky D.I., *Computer arithmetic in residual classes*, Soviet radio, Moscow, 1968 (in Russian).
[2] Chernyavsky A.F. (red.), *High-speed methods and systems of digital information processing*, Belarusian State University Press, Minsk, 1996 (in Russian).
[3] Kolyada A.A., Pak I.T., *Modular structures of pipeline digital information processing,* University Press, Minsk, 1992 (in Russian).
[4] Mohan P.V. Ananda, *Residue number systems: Algorithms and architectures*, Kluwer Academic Publishers, 2002.
[5] Omondi A., Premkumar B., *Residue number systems. Theory and implementation*, Imperial College Press, London, 2007.
[6] Vinogradov I.M., *Fundamentals of number theory*, Nauka, Moscow, 1981 (in Russian).