PRACE NAUKOWE Akademii im. Jana Długosza w Częstochowie
Technika, Informatyka, Inżynieria Bezpieczeństwa 2017, t. V, s. 147–153

http://dx.doi.org/10.16926/tiib.2017.05.12

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 MODULAR PRINCIPLES OF PARALLEL PIPELINE INFORMATION PROCESSING

Abstract. In the present paper, we deal with the methodology of implementation of the modular arithmetic algorithms using the parallel-pipeline residues summation blocks with respect to the bases of modular number system. These summation blocks are the main structural elements of high-speed modular operating devices, they provide high throughput performance of input data sets and are oriented to the wide application of VLSI chips.
Keywords: Modular number system, modular arithmetic, modular computing structures, parallel data processing, pipeline mode.

MODULARNE ZASADY RÓWNOLEGŁE POTOKOWEGO PRZETWARZANIA INFORMACJI

Streszczenie. W niniejszym artykule omówiono metodologię implementacji algorytmów arytmetyki modularnej przy wykorzystaniu równolegle potokowych bloków sumowania reszt w odniesieniu do podstaw modularnego systemu liczbowego. Te bloki sumujące są głównymi elementami strukturalnymi wysokowydajnych modularnych urządzeń obliczeniowych, one również zapewniają wysoką wydajność przepustowości zestawów danych wejściowych i są zorientowane na szerokie zastosowanie układów scalonych VLSI.
Słowa kluczowe: modularne systemy liczbowe, arytmetyka modularna, modularne struktury obliczeniowe, równoległe przetwarzanie informacji, tryb potokowy.

Introduction

The unique property of the modular number systems (MNS) to perform a natural decomposition of the 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 mapping of the computational processes into the high-speed parallel pipeline architectures [1-5].

The term "parallelism" is usually used to denote all the forms of simultaneous processing of information in a computer. Nevertheless, this concept is often interpreted in a more narrow sense confining itself to considering only those forms of processing that are characterized by the independence of simultaneously operating devices, systems, and their constituent parts. In this case, we can say about the spatial distribution of time overlapping computing processes or their components.

In contrast, the pipeline information processing is characterized by a spatio-temporal distribution of the implemented computational processes, i.e. by such computational organization at which each running process is distributed sequentially in time between all the used devices, systems or their elements. Both parallelism and pipelining can be considered at three logical levels: elementary operations on words, arithmetic operations and computation processes.

The modular computing structures (MCS) are focused on the high-speed processing of digital information on the first two of these levels [2-5]. The most significant feature of MA algorithms is that their implementation actually reduces to independent of each other summation operations of the sets of low-bit residues with respect to the modules of the number system. These operations are easily parallelized and they provide a high performance of the MCS when organizing the pipeline mode of information processing.

The basic notation and terminology

Let us introduce the following notation:

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

$\lfloor x \rfloor$ denotes the floor of x, i.e. the greatest integer less than or equal to x, $\lfloor x \rfloor = \max\{y \in \mathbf{Z} \mid y \leq x\}$;

$\lceil x \rceil$ 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 $m > 1$;

$|x|_m$ 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 generalized structure of algorithms of MNS-based computer arithmetic

The classical MNS is defined by means of a set of pairwise prime natural numbers (modules) $m_1, m_2, \ldots, m_k$ $(k > 1)$ by a mapping that assigns to each integer X $(X \in \mathbf{Z})$ a codeword of the form $(\chi_1, \chi_2, \ldots, \chi_k)$, where $\chi_i = |X|_{m_i}$ $(i = 1, 2, \ldots, k)$ [2, 3]. Concerning the computer algorithms of MA, it is easy to see that due to the internal parallelism of MNS each of these algorithms can in general be divided into k sections, the ith of which corresponds to the basic module m_i (i = 1, 2, …, k) and represents a certain set of operations on low-bit values. Such operations form the basis of the information processing at the level of elementary operations on words in the modular computing devices. In view of the low digit capacity of the operands, all the basic operations can be implemented in tabular way at the same time.

The analysis of algorithms of the MNS computer arithmetic shows that their sections have the same generalized structure, the typical element of which is a sequence of operations shown in Fig. 1.

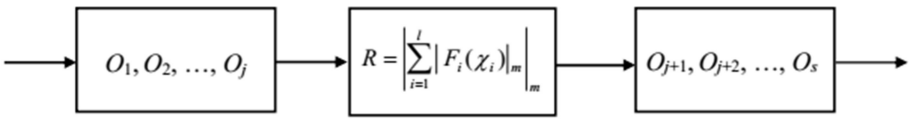

Fig. 1. A typical element of the section of MA algorithms

In general, this sequence contains two chains of basic operations $O_1, O_2, \ldots, O_j$ and $O_{j+1}, O_{j+2}, \ldots, O_s$ $(j > 0, s > j)$ on low-bit values, which are separated by the operation of a modular sum calculation of the form

$$R = \left| \sum_{i=1}^{l} |F_i(\chi_i)|_m \right|_m \qquad (1)$$

with the formation of the number of modulo overflows:

$$\Pi = \left\lfloor \frac{1}{m} \sum_{i=1}^{l} |F_i(\chi_i)|_m \right\rfloor, \qquad (2)$$

where $F_i(\chi_i)$ is the integer-valued function of residue χ_i with respect to some module m corresponding to the given section of the considered algorithm; $l \geq 2$.

The expression (1) and so the entire sequence of operations shown in Fig. 1 can easily be pipelined. Therefore, the MA algorithms have in general the same property. In other words, they have a pipeline structure.

The constructing principles of modular operating devices

The calculation of expressions of the form (1), (2) in the pipeline mode is most simply and efficiently carried out using the parallel-pipeline residues summation block (RSB), which structure is shown in Fig. 2.

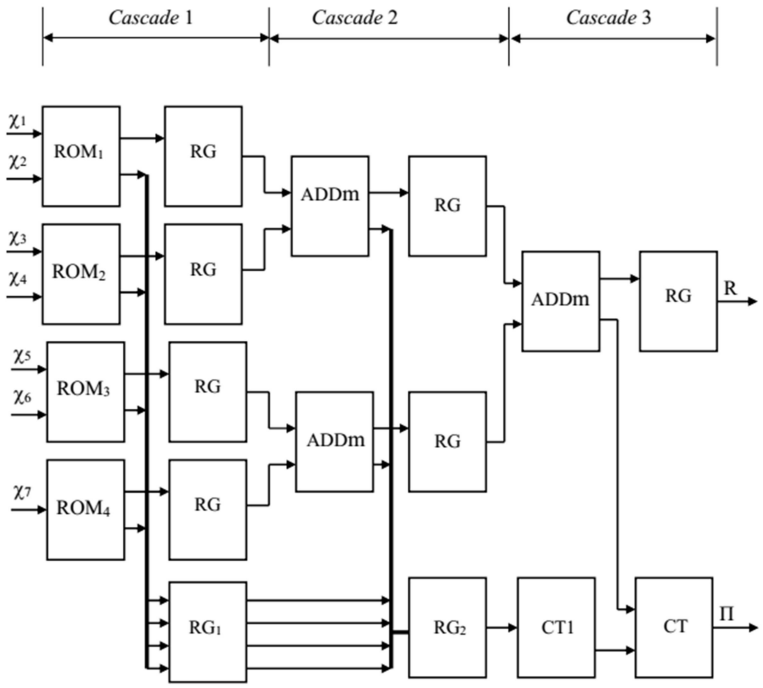

Fig. 2. The structure of the parallel-pipeline RSB (the case $l = 7$)

The parallel-pipeline RSB has a T_l-cascade pyramidal structure ($T_l = \lceil \log_2 l \rceil$) and is implemented with the help of the read only memory (ROM) ROM_j (j = 1, 2, …, t; $t = \lceil l/2 \rceil$), the adders *ADDm* modulo m with the overflow flag, the unit counter *CT*1; the registers *RG* of the cascades from the 1st to the T_lth, the registers RG_s ($s = 1, 2, ..., T_l - 1$) and the counter *CT*. The digit capacity of the registers *RG* is $b = \lceil \log_2 m \rceil$ bits, the counter *CT* is a $\lceil \log_2 (l-1) \rceil$-bit counter.

If the number l is even, then all the ROMs ($ROM_1, ROM_2, ..., ROM_t$) used in the first cascade of the RSB perform transformations of the pairs of input residues $\langle \chi_1, \chi_2 \rangle$, $\langle \chi_3, \chi_4 \rangle$, …, $\langle \chi_{l-1}, \chi_l \rangle$ into the pairs of values $\langle R_1, \omega_1 \rangle$, $\langle R_2, \omega_2 \rangle$, …, $\langle R_t, \omega_t \rangle$, respectively, where

$$R_j = \left| \left| F_{2j-1}(\chi_{2j-1)} \right|_m + \left| F_{2j}(\chi_{2j)} \right|_m \right|_m; \tag{3}$$

$$\omega_j = \left\lfloor \frac{1}{m} \left(\left| F_{2j-1}(\chi_{2j-1)} \right|_m + \left| F_{2j}(\chi_{2j)} \right|_m \right) \right\rfloor \tag{4}$$

(j = 1, 2, …, t).

As it follows from (3), ROM_j has a capacity of $2^{b_{2j-1}+b_{2j}}$ words of length $b + 1$ bits (b_{2j-1} and b_{2j} are the digit capacity of the residues χ_{2j-1} and χ_{2j}, respectively). The pair of values $\langle R_j, \omega_j \rangle$ is written to the ROM ROM_j at the address $\chi_{2j-1} + \chi_{2j} 2^{b_{2j-1}}$ for all possible values of the variables χ_{2j-1} and χ_{2j}. Thus, χ_{2j-1} and χ_{2j} are the lower and upper parts of the address, respectively, along which the desired two-dimensional value $\langle R_j, \omega_j \rangle$ is written to the ROM. For odd l, $ROM_1, ROM_2, \ldots, ROM_{t-1}$ transform the pairs of residues as described above, and ROM_t transforms a single residue χ_l into the value

$$R_t = |F_l(\chi_l)|_m . \tag{5}$$

Just this case is shown in Fig. 2. For odd l, ROM_t has a capacity of 2^{b_l} words of length b bits, and the value R_t is written to its memory at address χ_l for all the possible values of the variable χ_l.

The counter *CT*1 of units in the word serves for counting the number of unit bits in the binary code $(x_{l-3}\, x_{l-4} \ldots x_0)_2$ arriving from the output of the register RG_{T_l-1} in each clock cycle of the RSB. This function is implemented most simply using a ROM with a capacity of 2^{l-2} words of length T_{l-1} bits, and the value $\sum_{s=0}^{l-3} x_s$ is written to its memory at the address $\sum_{s=0}^{l-3} x_s 2^s$ for all

possible values of the variables x_0, x_1, …, x_{l-3} ($x_s \in \{0, 1\}$, $s = 0, 1, \ldots, l-3$). The counter *CT*1 can also be implemented as a special circuit triggered during one modular clock cycle.

In the RSB of the concerned type, the calculation of the modular sum R together with the formation of the overflow number Π (see (1) and (2)) takes T_l clock cycles. In the first clock cycle, the residues R_1, R_2, …, R_t (see (3) and (5)) are evaluated by means of $ROM_1, ROM_2, \ldots, ROM_t$ and are written into the corresponding registers RG of the first cascade, and the binary flags ω_1, ω_2, …, ω_{l_1} (see (4), $l_1 = \lfloor l/2 \rfloor$) are also stored in the l_1-bit register RG_1.

During the ν-th clock cycle (ν = 2, 3, …, T_l–1) the l_ν modular adders *ADDm* of the ν-th cascade ($l_\nu = \lfloor t_{\nu-1} \rfloor$; $t_j = \lfloor t_{j-1}/2 \rfloor$, j = 2, 3, …, $\nu - 1$, $t_1 = t$) add modulo m the pairs of residues obtained at the previous $(\nu - 1)$-th clock cycle. As this takes place, the generated overflow digits $\omega_{L_{\nu-1}+1}$, $\omega_{L_{\nu-1}+2}$, …, ω_{L_ν} ($L_\nu = L_{\nu-1} + l_\nu$, $L_1 = l_1$) are written to the group of l_ν upper bits of the register RG_ν. At the same time, the binary flags ω_1, ω_2, …, $\omega_{L_{\nu-1}}$ are passed from the register $RG_{\nu-1}$ to the group of $L_{\nu-1}$ lower bits of the register RG_ν. Let us note that in the case of odd $t_{\nu-1}$ the last $t_{\nu-1}$-th residue of the residue set passing to the ν-th cascade of RSB does not take part in paired summing, so it is passed into the last t_ν-th register *RG* of the ν-th cascade.

At the final T_l-th clock cycle of the residue summation process the adder *ADDm* of the T_l-th cascade evaluates the desired value of the modulo sum R (see (1)), the unit counter *CT*1 determines the number $\sum_{s=0}^{l-2} \omega_s$ of overflows modulo m occurred during all the previous clock cycles, which pass to the T_l-bit counter *CT* through its information input. The overflow digit ω_{l-1} generated by the adder *ADDm* is fed to the counting input of the counter *CT* and thereafter the desired value Π (see (2)) is formed in it.

For construction of this RSB the $l + 1$ ROMs, the $l + T_l - 3$ registers of the length b bit, the $T_l - 1$ registers of the length from l_1 to $l - 2$ bits and the counter are required. It is assumed that the modular adders *ADDm* and the unit counter *CT*1 are implemented by a ROM.

Due to the pipeline structure of the described RSB, the input data sets $\langle \chi_1, \chi_2, \ldots, \chi_l \rangle$ can enter to it in every clock cycle, i.e. with a frequency $f_{\text{MT}} = 1 / t_{\text{MT}}$, where t_{MT} is the duration of the modular clock cycle. Therefore, in the pipeline mode the streams of typical operation sequences will also be executed with the same speed (see fig. 1). Thus, the parallel-pipelined RSBs with respect to the modules m_1, m_2, ..., m_k of the number system, providing the maximum throughput performance for digital information processing at the level of operations on low-bit values, constitute the bulk of the equipment of high-speed modular operating devices.

The most important distinguishing feature of MA computer algorithms is that within the framework of pipeline structures of the considered type the simultaneous implementation of typical sequences of operations is easily performed even when such operations form the cycles. As a result, despite the sequential character of the applied procedures, in these cases the high-speed performance is achieved at relatively small hardware expenses. This circumstance also determines the simplicity of controlling the pipeline modular devices.

Concusion

The basic operation of all the non-modular procedures in the MA is the summation of the sets of residues with respect to the MNS modules together with the generation of the number of overflows occurred during their summation. These calculations are most simply and efficiently carried out by means of the parallel pipeline RSBs having a cascade pyramidal structure and processing the input data sets in every clock cycle. The parallel pipeline RSBs provide the highest speed for both parallel and serial implementations of non-modular procedures.

The fact that the performance of any operation on an arbitrary set of residues with a relatively small total digit capacity can always be carried out, in principle, in the tabular way by means of the same functional units (memory blocks, programmable logical matrixes, etc.) is of particular interest.

Thus, due to the internal parallelism, tabular structure and simplicity of pipelining at the level of operations on low-bit values, the MA algorithms represent an ideal basis for the synthesis of parallel pipeline modular computing devices, which provide high throughput performance and are oriented to the wide application of VLSI chips.

References

[1] Akushsky I.J., Yuditsky D.I., *Computer arithmetic in residual classes*, Soviet radio, Moscow, 1968 (in Russian).

[2] Chernyavsky A.F., Danilevich V.V., Kolyada A.A., Selyaninov M.Y., *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.