https://doi.org/10.31891/csit-2024-1-6
UDC 004.31:004.27

Volodymyr KHOROSHKO, Serhii ZYBIN
National Aviation University

METHOD OF ORGANIZING INTER-MODULE DATA EXCHANGE

The purpose of this article is the development and description of the method of organizing inter-module data exchange, that is based on the parametric presented subsystems, as well as the evaluation of the effectiveness of the developed multiprocessor computing systems. The authors propose a method of organizing inter-modular exchanges, which is based on the parametric presented subsystem. When developing programs using parallelization algorithms, parameters are used that set specific values for the subsystem. These values are determined during the formation of the subsystem. This makes the program independent of the graph of subsystem inter-module connections, on the basis of which this program will be implemented. The effectiveness of the computer system depends on a set of characteristics, which include: performance, reliability, probability of results, stability, manufacturability and others. To compare computer systems as a criterion of efficiency, it is necessary to adopt a comprehensive indicator that will take into account the main indicators. The real efficiency of a computer system is determined by the time it fulfils its functional purpose. Accordingly, the more reliable the system, the more relative time it is used for its intended purpose and the greater its efficiency. Thus, when comparing computer systems, it is necessary to compare their performance taking into account the real reliability of the system. At the same time, depending on the classes of problems being solved, the influence of various indicators (including reliability) on the efficiency of the system changes. Based on the results of the research, the following conclusions can be drawn: the most effective are the computer systems that are implemented on the developed MO. This fact is clearly visible from the comparison of the time spent on calculating not only the test task, but also when calculating the test equation with the test parameters.

Keywords: multiprocessor, inter-module, programmable structure, algorithm, parallelization, data exchange

Володимир ХОРОШКО, Сергій ЗИБІН
Національний авіаційний університет

МЕТОД ОРГАНІЗАЦІЇ МІЖМОДУЛЬНОГО ОБМІНУ ДАНИМИ

Метою даної статті являється розробка і опис методу організації міжмодульного обміну даними, який засновано на параметричному представлені підсистем, а також оцінювання ефективності розробленої мультипроцесорної обчислювальної системи.

Авторами пропонується метод організації міжмодульних обмінів, який заснований на параметричному представлені підсистеми. При розробці програм з використанням алгоритмів розпаралелювання використовуються параметри, які задають підсистему конкретними значеннями, які визначаються при формуванні підсистеми. Це робить програму незалежною від графа міжмодульних зв'язків підсистеми, на базисі якої ця програма буде реалізована.

Ефективність обчислювальної системи залежить від комплексу характеристик, до якого входять: продуктивність, надійність, ймовірність результатів, стійкість, технологічність та інші. Для порівняння обчислювальних систем у якості критерія ефективності необхідно прийняти комплексний показник, який буде враховувати основні показники. На жаль, у теперішній час немає подібного загальноприйнятого критерія ефективності обчислювальної системи.

Реальна ефективність обчислювальної системи визначається часом виконання свого функціонального призначення. Відповідно, чим надійніше система, тим більше відносного часу вона використовується за призначенням та, тим більша її ефективність. Таким чином, при порівнянні обчислювальних систем необхідно порівнювати їх продуктивності з урахуванням реальної надійності системи. При цьому, в залежності від класів задач, які вирішуються змінюється вплив різних показників (у тому числі надійності) на ефективність системи.

За результатами проведених досліджень можна зробити наступні висновки, найбільш ефективними є обчислювальні системи, які реалізовані на розробленому обчислювальному модулі. Цей факт наглядно видно з порівняння часу, який було витрачено на обчислення не тільки тестового завдання, але й при обчисленні тестового рівняння із тестовими параметрами.

Ключові слова: мультипроцесор, міжмодульна взаємодія, програмована структура, алгоритм, розпаралелювання, обмін даними

Introduction

Considering the history of the development of computing equipment, it can be seen that the increase in productivity and reliability of computer systems took place in two ways. The first way is the improvement of hardware. This happens mainly due to increasing the speed and reliability of the element base. The second way is the implementation of new architectures and principles of arrangement of computing processes.

Further improvement of the element base is based on the improvement of its production technology, which is approaching its physical limit.

Taking this into account, the second way is the most promising and relevant in terms of improving the productivity and reliability of computer systems.

All the variety of ways to build computer systems can be reduced to several methods of functional arrangement, which practically cover all existing structures of computer systems.

The development and creation of multicomputer and multiprocessor systems is an example of the emergence of new architectural solutions and principles of computing organization. Their construction is based on three principles: parallel execution of operations, variability of the structure, and structural homogeneity.

It is the combination of advanced concepts such as distribution and mutual influence at all levels that characterizes the hardware and mathematical support required to obtain a multiprocessor system. It is fundamentally important that the entire system functions under the guidance of a single operating system that arranges the information processing process.

At first glance, it may seem that by creating a multiprocessor computing system (MPCS), you can solve all the issues related to increasing productivity. However, there are two main problems that significantly complicate the solution of this problem:

• arrangement of connections between functional blocks of the system;

• arrangement of the computing process in the system.

The presence of these two problems during the creation of MPCSs requires looking for options for their construction and organization of processes, which, with some limitations, would ensure, on the one hand, the simplification of the system, and on the other hand, would not reduce efficiency much. This has led to several structural developments (constructions) of MPCS and several ways of organizing computing processes [1, 2, 3]. The key to the classification of MPCSs is the system of interrelationships, organization and functioning of the system.

The organization of the computing process in MPCS is the most perfect if all system resources, both hardware and software, are used efficiently, and the system's performance is as high as possible. However, such an organization requires a lot of efforts to implement. In order to reduce these efforts, some simplifications are sometimes made.

It is necessary to distinguish between two modes of operation depending on the performance of the MPCS and its dependence on the number of computing modules that are included in the system. Processors, single-chip micro-computers can be used as modules.

In the first case, sometimes large flows of relatively small tasks are processed in MPCS and the total performance of MPCS is close to the sum of the performances of the computing modules that are part of it. In the second case, which is the most important from the point of view of MPCS efficiency, the system solves one big problem, while each computing module solves some part of it. The necessary exchange of information is carried out between the modules in order to coordinate the entire process. In this case, large system losses appear on the operation of the operating system.

Currently, the development of MPCSs is one of the important and relevant directions of improving the productivity of computing devices.

The purpose of this article is the development and description of the method of organizing inter-module data exchange, that is based on the parametric presented subsystems, as well as the evaluation of the effectiveness of the developed MPCSs.

Development of the method of organizing inter-module data exchange

Subsystems are used for parallel solution of tasks in MPCS with a programmable structure [4, 5]. The subsystem $\boldsymbol{\theta}$ consists of the number n, where $n = |\theta|$, of computing modules (CM) required to solve the problem.

These modules are connected by channels so that it is possible to transmit data from any CM_i, $i \in (1, \dots, n-1)$ to all other CMs of the subsystem. In fig. 1 shows an example of some subsystem that consists of six CMs in the case of (N,4,4) graph of inter-computer system connections [5].

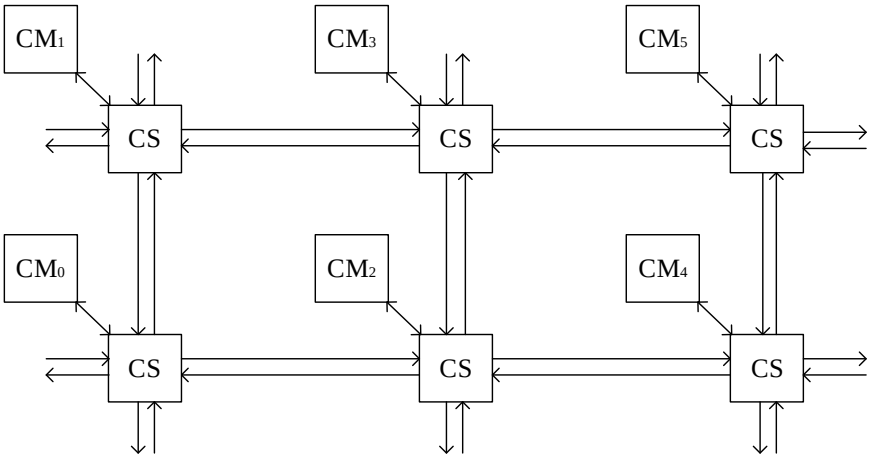

Fig. 1. The example of a subsystem consisting of six MOs. CM – computing module; CS – connection switch

Let the input and output streams of CM be labeled $p_i, p \in \{1, \dots, v\}$, where v is the degree of vertices of the graph of inter-module connections. It should be taken into account that CMs communicate with each other using a connection switch, which is implemented as a multi-channel interface [1]. Moreover, if CM_i and CM_j are connected by a channel (i_p, i_p), then these modules are also connected by channels (j_q, j_p), where (j_p, j_p) is a channel that is

connected to *p*-th output pole CM_i and to the *q*-th input pole CM_j. The channel (i_p, j_q) has the output queue OUT_p^i in CM_i, and the input queue IN_q^j in CM_j. The functioning of the channel (i_p, j_q) is that it transfers the data element from the beginning of the queue OUT_p^i to the end of the queue IN_q^j. This transfer is carried out in such a way that it does not require any period of time for its implementation. The data element, regardless of its length, instantly disappears from the OUT_p^i queue and appears in the IN_q^j queue (non-distribution of the data element). The operation of the channel (i_p, j_q) is activated if there is a data element in the queue OUT_p^i, and there is a free place in the queue IN_q^j for accepting its data element [4, 6].

In the process of solving the problem, the necessary exchanges are carried out between the modules of the subsystem, which are determined by the algorithm for solving the problem. Conveyors for inter-module exchanges are formed from channels and queues IN_p^i and OUT_p^i of computing modules CM_i. These modules are included in the subsystem $\boldsymbol{\theta}$, which solves the problem where $i = 0,1, \dots, |\theta| - 1, p = 0,1, \dots, v$; IN_0^i and OUT_0^i are queues that are located in CM_i. These queues contain the corresponding data that is intended to be transmitted from CM_i and the data that CM_i has received from other modules of the subsystem. Since the channel operation algorithm is fixed, the programming of inter-module exchanges (programming of the system structure) is reduced to the task of the discipline of transferring data elements between queues of each internal CM. Thus, in the subsystem shown in fig. 1, the following actions are performed in CM_2. Data arriving at the input of queue IN_4^2 from CM_0 are transferred to queues OUT_1^2, OUT_2^2, IN_0^2, and data from queues IN_1^2, OUT_0^2, IN_2^2 are forwarded to queue OUT_4^2 for transmission to CM_0.

That is, a method of organizing inter-modular exchanges is proposed, which is based on the parametric presented subsystem. When developing programs using parallelization algorithms, parameters are used that set specific values for the subsystem. These values are determined during the formation of the subsystem. This makes the program independent of the graph of subsystem inter-module connections, on the basis of which this program will be implemented. In the following, we will call the parametric representation of subsystems the environment, and the individual parameters of this representation – elements of the environment.

Description of the method

The proposed method is as follows.

1. The first stage of the proposed method.

As an environment in the subsystem $\boldsymbol{\theta}$ with the number of modules *n*, we select the addresses A_i, a number E_i, a path $G_i(A_i \oplus A_j)$, a translation $F_i(A_i \oplus A_j)$ and a weight T_i functions defined in each CM_i subsystem θ, where $\mathrm{i} = \mathrm{j} = 0,1, \dots, \mathrm{n} - 1$. The question of selecting the modules included in the $\boldsymbol{\theta}$ subsystem and forming their addresses, path and translation functions is described in [6]. In the future, the formation of the environment (using a decentralized algorithm) will not be considered. Attention will be paid to determining the values of the environment elements $A_i, E_i, G_i, F_i, T_i, i = 0,1, \dots, n - 1$, which are determined after the formation of the subsystem.

The organization of inter-module transmissions is based on the use of CM addresses. The transmission of the data block *Y* from the module $CM_i, i \in \{0,1, \dots, |\theta| - 1\}$ in some subsets of modules of the subsystem $\boldsymbol{\theta}$ occurs by broadcasting the switching message $H\{A_i, B_1, \dots, B_k, Y\}$ through sequence of subsystem modules $\boldsymbol{\theta}$. These modules are located between the transmitter module CM_i and the receiver module CM.

At the same time, when switching channels, the data block ***Y*** has service information that allows to switch the channel system from the transmitter module to the receiver module for the next data transfer [8, 9]. However, both channel switching, and message and packet switching use the same algorithms to control the transmission of switching messages. In what follows, we will not distinguish between methods of organizing data transmission and will assume that if a CM receives a switch message destined for it, then that CM has received the data sent to it.

The switching message $H\{A_i, B_1, \dots, B_k, Y\}$ consists of the following components. ***H*** is a switch message flag that indicates the type of path procedure used to transmit the switch message under consideration. B_j is an identifier that specifies the set of addresses of modules that should receive the data block ***Y***, $j = 1,2, \dots$. $k \in \{1,2, \dots\}$. A_i is an address of the module that generated the switching message under consideration. To define the entire set of subsystem modules $\boldsymbol{\theta}$, such as receivers (for translational exchange [10]), we will use the ***YCI*** identifier. The switching message $H\{A_i, YCI, Y\}$ is formed in CM_i and delivered from CM_i to all subsystem modules. This ensures that they accept the ***Y*** data block.

The CM addresses of the subsystem should be set so that the algorithm for determining the route of transmission of the switching message (route procedure [8, 10]) is easy. Let's consider one of these ways of addressing modules.

To determine the addressing in the subsystem, the module CM_0, which is called the main one, is allocated. Let ***m*** denote the distance from the main CM_0 to the farthest module of the subsystem. In the case of a subsystem, the connection graph shown in Fig. 2, m=3 (the furthest from CM_0 is CM_5).

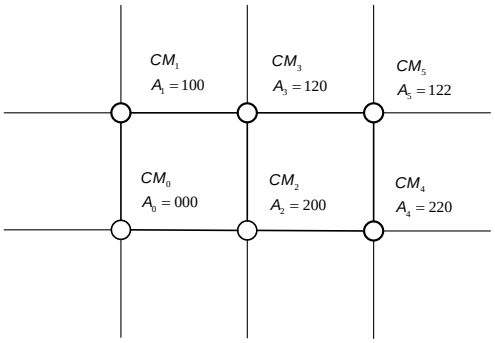

Fig. 2. An example of a subsystem to which the connection graph corresponds

Each $CM_i, i \in \{0,1,\dots,|\theta|-1\}$ subsystem θ assigns an address given by the vector A_i, $A_i = A_i^0 A_i^1 \dots A_i^{m-1}$, $A_i^j \in \{1,2,\dots,v\}, j = 1,2,\dots,m-1$, where v are the degrees of vertices of the graph of inter-module connections. The number $A_i^j, j \in \{1,2,\dots,m-1\}$ is called the row number of the *j*-th elementary module CM_i. The address A_0 of the main module CM_0 is set as $A_0^0 A_0^1 \dots A_0^{m-1} = 00\dots0$, i.e. $A_0 = 0$ for all $j \in \{1,2,\dots,m-1\}$. All other modules of the subsystem receive addresses that determine their position relative to the main module CM_0. Thus, the address $A_i, i \in \{0,1,\dots,|\theta|-1\}$ is a sequence of CM output pole numbers that specify the shortest path from the main module CM_0 to CM_i. If there are several paths of the same length between CM_0 and CM_i, then the sequence corresponding to one of the specified paths is selected as the address A_i.

At the same time, if the length from CM_0 to CM_i is equal to $e, e < m$, then at the address CM_i the tier numbers $A_i^e A_i^{e+1} \dots A_i^{m-1}$ are equal zero. Modules CM_i and CM_k are included in the address subsystem $R_r(A_i^0 \dots A_i^{r-1})$ of tier ***r***, if $A_i^j = A_i^k$ for all $j < r, r = 1,2,\dots m$. By definition, we assume that subsystem θ is an address subsystem of tier zero. The address subsystem $R_m(A_i^0 \dots A_i^{m-1})$ of tier ***m*** includes only one module $CM_i$with the address $A_i = A_i^0 A_i^1 \dots A_i^{m-1}$.

It is obvious that the modules of the address subsystem $R_r(A_i^0 \dots A_i^{r-1})$, $r = \overline{0,\dots,m}$ create a connected subgraph of the system structure as for any CM_i with the address $A_i = A_i^0 \dots A_i^{r-1}$=0…0 there is a path to any CM_k with the address $A_k = A_i^0 \dots A_i^{j-1} A_i^2 \dots A_k^{m-1}$. At the same time, the tier number A_k^r is equal to the output pole number CM_i, which is on the shortest path from CM_i to CM_k.

2. The second stage of the proposed method.

The addressing of subsystem modules $\theta \in D(\lceil \log_2(v+1) |, m)$ considered in paragraph 1 is addressing [8], and, therefore, for determining the shortest paths from CM_i, $i \in \{0,1,\dots,|\theta|-1\}$, the tier functions $G_i^r(A_i^r \oplus G_k^r)$ can be used. In this case, $\oplus$ is an addition operation modulo $(v+1)$, $\lceil x|$ is the smallest integer that is not less than ***x***.

The value of $G_i^r(A_i^r \oplus A_k^r)$ at $r = \min\{j / A_i^j \neq A_k^j, j = 0,\dots,m-1\}$ is equal to the output pole number CM_i, which belongs the shortest path from CM_i to the address subsystem $R_{r+1}(A_i^0 \dots A_i^{r-1} A_k^r)$.

For $r = \min\{j / A_i^j \neq A_k^j, j = 0,\dots,m-1\}$ the value of $G_i^r(A_i^r \oplus A_k^r)$ is undefined and corresponds to the empty set of poles $\varnothing$. At the same time, the shortest path between CM_i and the address subsystem $R_{r+1}(A_i^0 \dots A_i^{r-1} A_k^r)$ means the path with the length $d_{ih} = \min\{d_{ig} j / CM_g \in R_{r+1}(A_i^0 \dots A_i^{r-1} A_k^r)\}$ between CM_i and CM_h such that $CM_h \in R_{r+1}(A_i^0 \dots A_i^{r-1} A_k^r)$. The table value $G_i^r(X), r = 0,1,\dots,m-1, X = 0,1,\dots,v$ needs $(v+1)m\lceil \log_2(v+1) |$, of bits in each $CM_i, i \in \{0,1,\dots,|\theta|-1\}$. However, the sizes of the tables for storing the values of the tier functions $G_i^r(X), i = 0,1,\dots,|\theta|-1, X = 0,1,\dots,v$ can be reduced as a result of the addressing property used. Indeed, if in each CM_i we store the value of the distance t_i between CM_0 and CM_i, then the value of $G_i^n(A_i^r \oplus A_k^r)$ at $r \geq t_i, r = \min\{j / A_i^j \neq A_k^j, j = 0,\dots,m-1\}$ do not require the storage because $G_i^n(A_i^r \oplus A_k^r) = A_k^r$. Based on this, in CM_i, which is at a distance t_i from CM_0, it is necessary to have a table only for saving the values of the tier functions $G_i^r(X)$. That is, the size of the table CM_i should be $(v+1)t_i|\log_2(v+1)|$ bits.

In the case of subsystem $\boldsymbol{\theta}$ (Fig. 2), the values of the tier functions $G_i^0(X), G_i^1(X), G_i^2(X)$ are given in the table 1. Other values of layer functions that must be stored are not defined. This is due to the specificity of the subsystem $\boldsymbol{\theta}$.

Tier functions are set in such a way that for each CM_i, the system of paths from CM_i to all the last modules of the subsystem $\boldsymbol{\theta}$ form a spanning main tree [6] D_i of the subsystem , $i = 0,1,\dots,|\theta|-1$. Thus, for each $CM_k, k \neq i$, a single input pole is indicated, which lies on the path along the spanning tree D_i from the main module CM_i to CM_k. In fig. 3 shows spanning trees with the main $CM_i, i = 0,1,2,\dots,5$ for the subsystem whose connection graph is shown in fig. 3.

Table 1

Value of Tier Functions

CM_i	r		
	0	1	2
1	$G_1^0(1)=3$ $G_1^0(3)=3$	$G_1^1(1)=2$	0
2	$G_2^0(3)=1$ $G_2^0(2)=4$	$G_2^1(2)=2$	$G_2^2(2)=2$
3	$G_3^0(3)=3$ $G_3^0(1)=3$	$G_3^1(2)=4$	$G_3^2(2)=2$
4	$G_4^0(3)=1$ $G_4^0(2)=4$	$G_4^1(2)=4$	$G_4^2(2)=4$
5	$G_5^0(1)=3$ $G_5^0(3)=3$	$G_5^1(2)=4$	$G_5^2(2)=4$

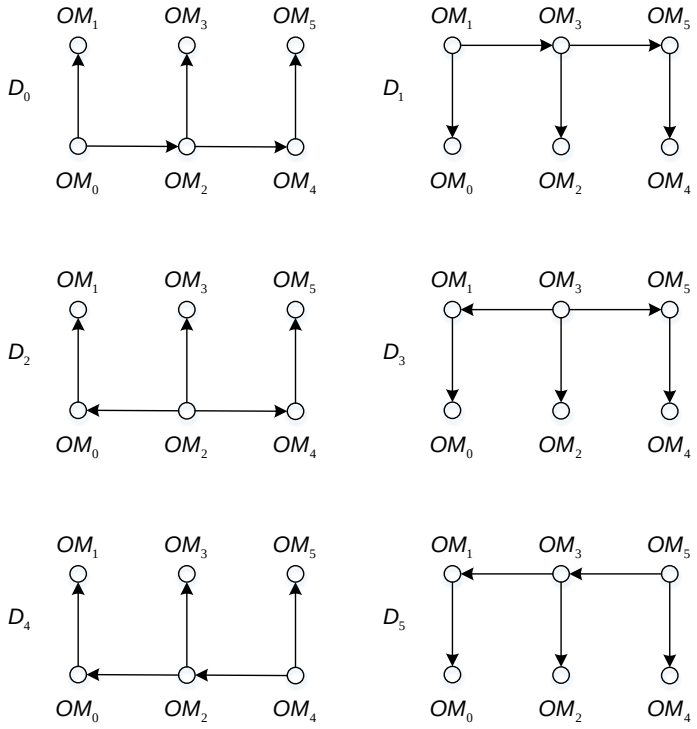

Fig. 3. The Connection Graph for the Subsystem

We will call the path function CM_i $G_i(A_i \oplus A_j) = \cup_{r=0}^{m-1}\{G_i^r(A_i^r \oplus A_j^r)\}, j \in \{0,1,\dots,|\theta|-1\}$, $A_i^r \oplus A_k^r$ – componentwise addition modulo $(v+1)$ of vectors A_i and A_j.

3. The third stage of the proposed method.

To organize transmissions during broadcast exchange, it is necessary to set the broadcast function $F_i(A_i \oplus A_j)$ in each CM_i. The value of the functions $F_i(A_i \oplus A_j)$ is the ordered set of output poles CM_i, through which the switching message $H\{A_j, BCI, Y\}$ enters the modules adjacent to CM_i. At the same time, the switching message $H\{A_j, BCI, Y\}$ must be distributed between modules of the subsystem along the main tree D_j, $j \in (0,1,\dots,|\theta|-1)$.

For the subsystem, the connection graph of which is shown in fig. 2 values of the translation function are not listed in the table. 2. Values of the translation function not listed in the table have the value ∅.

Table 2

Values of the translation function

CM_i	F_i				
	(000)	(100)	(200)	(020)	(002)
0	1,2	∅	1	∅	∅
1	2,3	∅	∅	3	∅
2	2,3,4	2	∅	2,3	3,4
3	3,4	∅	∅	3	3
4	1,2,4	∅	1,2	1,2	∅
5	1,4	∅	1	1	∅

4. The fourth first stage of the proposed method

The tree D_0 is used to number the modules of the subsystem $\boldsymbol{\theta}$ and to specify the weighting functions $T_i, i = 0,1, \dots n-1$. The vertex weight of the original covering tree is called [12] the number of vertices of the covering tree that have the vertex under consideration as a root. We denote the weight of vertex $\boldsymbol{i}$ by $\tau_i, i = 0,1, \dots |\theta| - 1$. So for the subsystem (Fig. 2) and the covering tree D_0, in Fig. 3 shows that the weights of the vertices have the following values: $\tau_0 = 6, \tau_1 = 1, \tau_2 = 4, \tau_3 = 1, \tau_4 = 2, \tau_5 = 1$. We will assign numbers E_1 to modules $CM_i, i = 0,1, \dots n-1$, according to the following algorithm.

Step 1. It is necessary to assign the number zero to the main module.

Step 2. It is necessary to find the numbered module CM_j with number $\boldsymbol{j}$, which is the main one for the original covering tree. The vertices of this tree have no numbers. If there is no such vertex, then we proceed to step 5.

Step 3. It is necessary to form a set $\boldsymbol{A}$ that includes connections with CM_j that do not have numbers. Then it is necessary to select the module from the set $\boldsymbol{A}$, which is connected to CM_j channel with the smallest label of the output pole. Next, it is necessary to assign the number $\boldsymbol{j+1}$ to the selected module and remove CM_{j+1} from the set $\boldsymbol{A}$, $\alpha = \tau_{j+1}$.

Step 4. If $A = \emptyset$, then it is necessary to go to step 2, otherwise, it is necessary to select the module from the set $\boldsymbol{A}$, which is connected to CM_j channel with the smallest output pole label. Then it is necessary to assign the number to the selected module $g = j + \alpha + 1$; $\alpha = \alpha + \tau_g$ and remove CM_g from the set $\boldsymbol{A}$. Finally, proceed to step 4.

Step 5. The end.

5. The fifth stage of the proposed method

The values of the weighting function T_i are the set $(\tau_{i_1}, \dots, \tau_{i_k})$ weights τ_{i_p} of modules $CM_{i_p}, p = 0,1, \dots, k$ arranged by growth of indices. Modules CM_{i_p} are connected with CM_i by channel (i_q, i_p), where $k = |F_i(A_i \oplus A_0)|, q \in F_i(A_i \oplus A_0), r = G_{i_p}(A_i \oplus A_0)$. Because, for the subsystem (Fig. 2) $F_2(A_2 \oplus A_0) = \{1,2\}, T_2 = \{\tau_3, \tau_4\} = \{1,2\}$; $F_0(A_0 \oplus A_0) = (1,2), T_0 = \{\tau_1, \tau_2\} = \{1,4\}$; $F_4(A_4 \oplus A_0) = \{1\}, T_4 = \{\tau_5\} = \{1\}$; $T_1 = T_3 = T_5 = \emptyset$.

6. The sixth stage of the proposed method

Thus, after building a subsystem $\boldsymbol{\theta}$ with a user-specified number of modules $\boldsymbol{n}$, the neighbourhood $\{A, G, E, F, T\}$ will be defined in each CM. The user knows that the subsystem has a module CM_0 with number $E_0 = 0$ and address $A_0 = 00 \dots 0$. If the user is not satisfied with the numbering and addressing of the modules adopted during the creation of the system, he can propose his own numbering and addressing that he needs. Loading tasks into the system consists in the fact that each CM receives the E_i program and data corresponding to its number. To organize differential inter-module exchanges [7], you can use the $CM_i, \mathrm{i} = 0,1, \dots, \mathrm{n}-1$ correspondences $j \Leftrightarrow A_j, \mathrm{j} = 0,1, \dots, \mathrm{n}-1$ between module numbers available in each module and their addresses. Correspondences are specified by entering correspondence tables $j \Leftrightarrow A$ into each module. If this method of matching is unacceptable and the specifics of the task do not solve the problem of processing speed, then it is necessary to remove differential exchanges from the program and replace them with translational ones. At the same time, the data issued by the CM must have the number of the module that issued it. In the receiving module, this number is used to set the order of

processing of incoming messages. Note that when sending a message to the CM_0 module, there are no difficulties in implementing differential exchange, because by agreement $A_0 = 00 \ldots 0$.

After the numbering and addressing of the modules is carried out in the $\boldsymbol{\theta}$ subsystem, its efficiency and productivity must be determined.

The effectiveness of the computer system depends on a set of characteristics, which include: performance, reliability, probability of results, stability, manufacturability and others [13]. To compare computer systems as a criterion of efficiency, it is necessary to adopt a comprehensive indicator that will take into account the main indicators. Unfortunately, at the present time, there is no similar generally accepted criterion for the efficiency of a computing system. The absence of such a single efficiency criterion is caused by significantly different requirements for the main characteristics.

A common criterion that is widely used to evaluate the performance of a computing system is the performance-to-cost ratio

$$EF = P/C \tag{1}$$

It is well known that with the development of the element base and computer technology, the P indicator increases. Its increase occurs due to the improvement of the characteristics of the element base and the architecture of the computer system. This indicator rather reflects the stage of development of the computing system than the individuality of the system.

The real efficiency of a computer system is determined by the time it fulfils its functional purpose. Accordingly, the more reliable the system, the more relative time it is used for its intended purpose and the greater its efficiency. Thus, when comparing computer systems, it is necessary to compare their performance taking into account the real reliability of the system. At the same time, depending on the classes of problems being solved, the influence of various indicators (including reliability) on the efficiency of the system changes.

As an efficiency criterion for comparing computer systems, we can use a generalized indicator of the form:

$$EF = (K_V P_0)/C \tag{2}$$

where P_0 is the performance of the computer system with ideal reliability; C – the cost; K_V is the coefficient of survival of the system.

The system survivability coefficient is the ratio of the number of states $\boldsymbol{S}$ of the system that correspond to operational efficiency to the entire set of states S_n^i (where $\boldsymbol{i}$ is the multiplicity of generalized failures, $\boldsymbol{n}$ is the number of computing systems)

$$K_V = S/S_n^i$$

When solving problems that require a lot of time and are compared with the time of failure, an essential indicator of reliability that determines efficiency is the probability of continuous solution of the problem $P_p(t)$. That is, the probability that during the time of solving the problem T_p the system will not fail. Since in this case the positive effect will appear only as a result of error-free execution of the task, then it can be written that the real performance P_r of the computer system when solving this class of tasks will be [14]

$$P_r = (K_V P_0\, P_p(t))/C \tag{3}$$

Thus, when comparing computing systems that are focused on solving problems that require a long solution time, we use the following expression as an efficiency criterion

$$EF = P_r = (K_V P_0\, P_p(t))/C \tag{4}$$

The criterion (4) is more generalized compared to (2). When solving problems in which $P_p(t) = 1$, criteria (2) and (4) coincide.

It should be noted that the indicators K_V and $P_p(t)$ are often used as independent indicators of the efficiency of computer systems. However, the performance indicator of the computing system, which is based only on the consideration of reliability, is one-sided.

The $\boldsymbol{EF}$ indicator is a product of traditional performance indicators, which is more sensitive than each of the components taken separately. It allows to evaluate a real efficiency and reduction of efficiency due to unreliability of systems. The proposed efficiency indicator does not contradict the well-known conclusions that not all structural methods of improving efficiency, for example, reliability, lead to an increase in the overall efficiency of the computing system. Increasing the speed or reliability of an element base cannot effectively increase $\boldsymbol{EF}$

because of the increased cost. Increasing performance at the expense of additional hardware leads to a decrease in reliability and an increase in cost.

The productivity of computing systems can be described by the following expression

$$P_0 = \frac{K_A}{t_f}\lambda_{max} t_i \tag{5}$$

where K_A is the coefficient that determines the architecture of the computing system;
λ_{max} is the intensity of exchange of informational messages;
t_i is the task execution time;
t_f is the operation execution time or information message generation time.

$$t_f = nk_l T_c + (\lambda_1 k_1 + \lambda_2 k_2) t_p$$

where k_l is the connection coefficient in the computing system;
T_c is the cycle execution time;
$\lambda_1 = 0{,}7$ and $\lambda_2 = 0{,}3$ are the relative coefficients of the use of short and long operations in computer systems. The use of a modern element base allows reducing the average values of the coefficients $0{,}7k_1$ and $0{,}3k_2$ to values of 30÷50;
t_p is the operation execution time.

$$k_l = k_x \frac{\sum_{i=1}^{p} \rho_A(i)}{\sum_{i=1}^{p} \rho_B(i)} \tag{6}$$

where $\sum_{i=1}^{p} \rho_A(i)$ is the number of units in the adjacency matrix $\boldsymbol{A}$ of the implemented algorithm;
$\sum_{i=1}^{p} \rho_B(i)$ is the number of units in the adjacency matrix $\boldsymbol{B}$ of the implemented algorithm;
k_x is the coefficient that takes into account the overlap of matrix $\boldsymbol{A}$ with matrix $\boldsymbol{B}$. If matrix $\boldsymbol{B}$ completely overlaps matrix $\boldsymbol{A}$, then $k_x = 1$, otherwise $k_x = 0$.

The data of the evaluation of the efficiency for the number of connections allow to evaluate the efficiency of the execution of the programs for the computer system and are determined as follows.

First, the organization of exchanges that accelerate the execution of programs requires an increase in the number of connections between CMs. However, the number of these connections is limited by the hardware implementation of the CM due to the fixed number of outputs of the integrated circuits on which they are implemented.

Second, complex information exchanges between CMs are used during program execution.

Regarding the choice of a CM architecture with an effective organization of connections, compatibility matrices are created to analyze the proposed architectures and compare them with each other. At the same time, the efficiency of connections is calculated. If $k_l = 100\%$, then the required architecture is found, if $k_l \neq 100\%$, then it is necessary to choose the architecture in which k_l has the largest value.

The impact of the CM architecture on performance is taken into account by the K_A factor. Increasing system performance by complicating the architecture and improving the principles of information exchange does not always give a positive result.

An important problem in the synthesis of MPCSs with a programmable structure is the organization of memory. Just like in MPCS, it can be organized by concentrated memory, branched memory or concentrated-branched memory.

Any direct communication channel between computing modules or between microprocessors in CMs can be programmed in MPCS with a programmable structure using a switching structure. However, this leads to the fact that in the system or in the CM, the conditions for exchanging information of the centralized memory with other elements are created only on the basis of the trunk principle.

The most promising in terms of speed and flexibility are multiprocessor systems with branched memory and universal switching. Linear, planar or spatial structures can be implemented in computing systems. That is, in systems with programmable switching, it is not necessary to provide rigid direct communication channels between microprocessors and elements of branched memory. There is an opportunity to connect elements of branched memory and communication structure instead. In the future, in the communication structure, it is necessary to organize, as necessary, direct communication channels between microprocessors and elements of branched memory by software. This makes the system more flexible. The general restructuring of communication channels qualitatively changes the structure of the MPCS, which allows programming the structure of the device and thus directly influencing the architecture of the computer system, and not only the process and result of information processing.

The further development of the communication structure is the creation of a spatial communication structure to solve the problems of building communication channels in three dimensions. In the general case, such a communication structure can be presented in the form of a cube on three adjacent faces of which there are information inputs, and on the other three adjacent faces there are three information outputs.

The CM can be implemented on VLSI (Very Large Scale Integration), and, depending on the tasks that the module solves, its architecture can change. CM is a hardware and software unit of the system, which implements the main functions (for which it was created), as well as additional ones (within the area of problem orientation). The CM provides a change of connections with other modules and a change of its mode of operation during reconfiguration of the computing system. The introduction of the zone of problem orientation is due to the fact that a module has been created in MPCS with a programmable structure, which implements all the functions of the system. This module is included due to high total redundancy. Therefore, all the performed functions are divided into groups, and each of them defines a zone of problematic orientation, within which the module is configured by replacing the program. The number of groups is determined by the number of module types.

Due to the fact that the CM includes several microprocessors and different types of memory, these modules can be considered as a computing system for collective use. For its use, a computer system model is used, which is called a "multi-resource" mass service system."

Taking into account the above, we will analyse the proposed CM architectures, which differ from each other in the structure of the memory implementation module. In addition, memory is a distributed resource because communication is carried out through a common bus. A management of information exchange is carried out in accordance with the principles of interface construction. Before the start of information transfer, the active microprocessor determines the readiness of the general bus and further determines the readiness of the specified resource in receiving information.

As noted earlier, the general memory can be divided into local modules. That is, the proposed CMs have the following architecture: CM_1 (Fig. 4), CM_2 (Fig. 5) and CM_3 (Fig. 6) [1, 15, 16]. The communication structure is implemented accordingly.

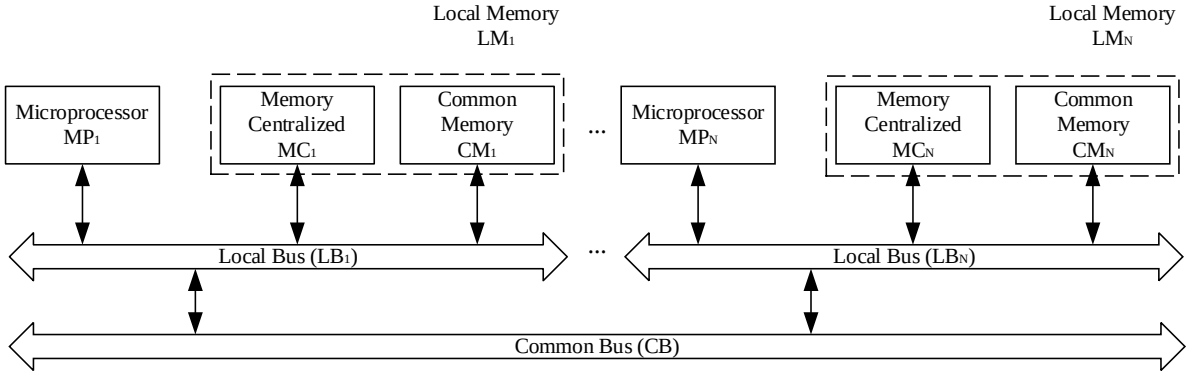

Fig. 4. The Architecture of CM_1

In order to evaluate the technical capabilities of CMs and MPCSs with a programmable structure (Fig. 1), a comparative evaluation has carried out using test tasks. When evaluating CMs, 2, 5, and 8 processors have used in the module. The results of the test evaluation are given in the table 3, 4, 5. The 2x2 and 3x3 matrix has used to evaluate the computer system.

Single-board micro-computers were installed in the nodes of the 3x3 matrix. The development of MPCS with a programmable structure was performed according to the structure of a 2x2 matrix, in the nodes of which CM_1 and CM_3 were used. The obtained results are shown in the table 6 and table 7.

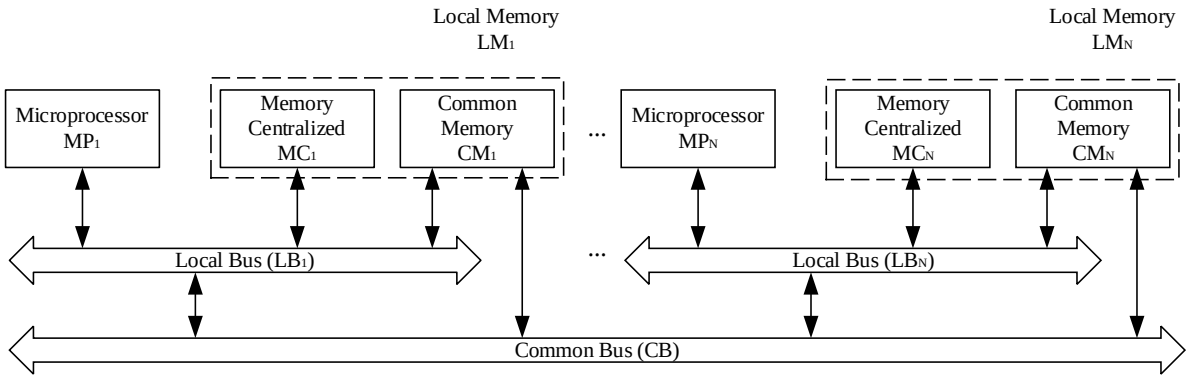

Fig. 5. The Architecture of CM_2

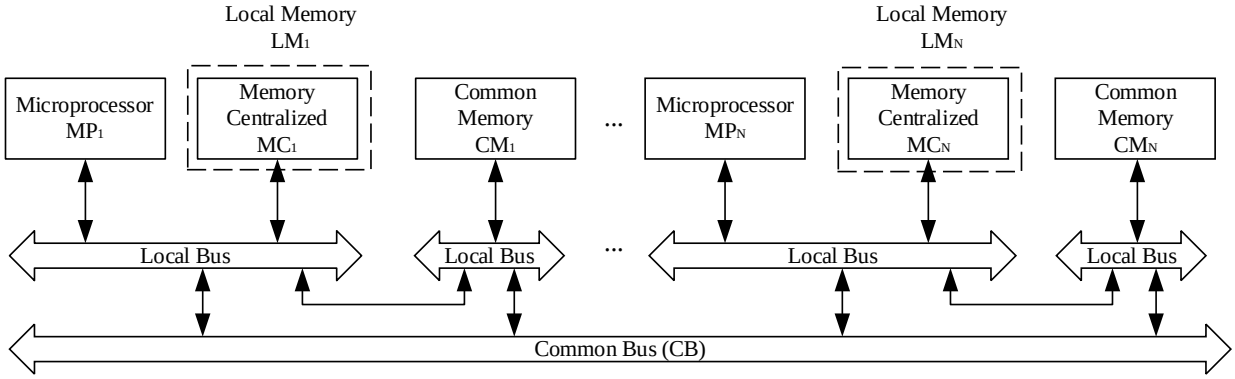

Fig. 6. The Architecture of CM_3

The construction and technological equipment CM has two construction options:

a) an option that allows implementation in a hybrid version without the use of cases;
b) variant in a monolithic polymer case, in the form of a hermetic micro assembly.

The second option ensures maintainability, because it allows to remove and brew the cover several times. All microcircuits included in micro blocks are caseless. Multi-level interfaces and framing of microprocessors, RAM and non-volatile memory are implemented on multi-matrix crystals, which makes it possible to obtain a high density of assembly with minimal dimensions of integrated circuits.

Table 3

Productivity of a dual-processor system

The amount of load on the processor connections	CM_1	CM_2	CM_3
0.0	1.00	1.00	1.00
0.2	0.63	0.70	0.68
0.4	0.48	0.64	0.50
0.6	0.36	0.44	0.39
0.8	0.30	0.37	0.32
1.0	0.26	0.31	0.27

Table 4

Performance of a five-processor system

The amount of load on the processor connections	CM_1	CM_2	CM_3
0.0	1.00	1.00	1.00
0.2	0.58	0.65	0.62
0.4	0.39	0.42	0.41
0.6	0.29	0.31	0.30
0.8	0.23	0.24	0.24
1.0	0.19	0.20	0.19

Table 5

Performance of an eight-processor system

The amount of load on the processor connections	CM_1	CM_2	CM_3
0.0	1.00	1.00	1.00
0.2	0.49	0.51	0.50
0.4	0.25	0.25	0.25
0.6	0.17	0.17	0.17
0.8	0.12	0.12	0.12
1.0	0.10	0.10	0.10

Table 6

Results of the Assessment

The structure of the computing system	**Test task execution time, ms**	**Relative time consumption, %**
The matrix 3x3 with microcomputer at nodes	867	100
The matrix 2x2 with MO_1 at nodes	385	44,4
The matrix 2x2 with MO_3 at nodes	329	37,9

Table 7

Results of the Assessment

	Computing Type				
	One	Three	CM_1	CM_2	CM_3
The time spent on the calculation of one test parameter, ms	1.2	0.54	0.35	0.31	0.33
Time spent on solving the test equation, ms	5.2	1.9	1.23	1.085	1.155

Conclusion

Taking into account the above, we have analysed the proposed architectures, which differ from each other in the structure of the memory implementation module. In addition, memory is a distributed resource because communication is carried out through a common bus. A management of information exchange is carried out in accordance with the principles of interface construction. Before the start of information transfer, the active microprocessor determines the readiness of the general bus and further determines the readiness of the specified resource in receiving information.

Based on the results of the conducted research, conclusions can be drawn. The most effective are computing systems that are implemented on the developed computing modules. This fact is clearly visible from the comparison of the time spent on calculating not only the test task, but also when calculating the test equation with the test parameters.

References

1. Michael Hübner. Jürgen Becker. Multiprocessor System-on-Chip - Hardware Design and Tool Integration. 2011. P. 280.
2. Ivan Opirskyy, Serhii Zybin, Vladimir Horoshko. Analysis of Mathematical Models of Functioning Scalar Multiprocessor Systems. // INFORMATION SYSTEMS AND NETWORKS. Volume 6, 2019. pp. 66 – 78.
3. V. KHOROSHKO, M. BRAILOVSKYI, M. KAPUSTIAN. Multi-criteria Assessment of the Correctness of Decision-making in Information Security Tasks// International Scientific Journal "Computer Systems and Information Technologies". 2023. # 4. PP. 81-86.
4. A.P. Godse, D.A. Godse. Microprocessor and Microcontroller System. Technical Publicationa Pune. 2007. P. 644.
5. D. Zeng. Future Intelligent Information Systems: Volume 1. Springer-Verlag Berlin. 2016. P. 689.
6. G**.** Kornaros**.** Multi-Core Embedded Systems. CRC Press**.** Taylor & Francis Group. 2010. P. 493.
7. Serhii Zybin, Vladimir Khoroshko, Volodymyr Maksymovych, Ivan Opirskyy. Effective Distribution of Tasks in Multiprocessor and Multi-Computers Distributed Homogeneous Systems. // International Journal of Computing. VOLUME 20(2), 2021, pp. 211 – 220.
8. *S.V. Zybin, V.O. Khoroshko. Performance and optimization of specialized information processing systems with a software-configurable structure // Informatics and mathematical methods in modeling - Odesa - ONPU, Vol. 9 - No. 3 - 2019 - P. 120 - 130.*
9. D. OKRUSHKO, A. KASHTALIAN. System of Distribution and Evaluation of Tasks in the Software Development Process // International Scientific Journal "Computer Systems and Information Technologies". 2023. # 2. PP. 86-97.
10. John L. Hennessy. David A. Patterson. Computer Architecture, Sixth Edition: A Quantitative Approach. ELSEVIER INDIA. 2017.
11. Andrew S. Tanenbaum, Todd Austin. Structured computer organization. Sixth edition. Pearson. 2013. P. 801.
12. Douglas West . Introduction to Graph Theory, 2nd edition. Pearson Modern Classic. 2017.
13. John Paul Shen. Mikko H. Lipasti. *Fundamentals of Superscalar Processors*. WAVELAND. PRESS, INC. 2013. P. 658.
14. A. Elahi. Computer Systems: Digital Design, Fundamentals of Computer Architecture and Assembly Language. Springer International Publishing. 2018. P. 269.
15. Andrew S. Tanenbaum, Maarten Van Steen. Distributed Systems Principles and Paradigms. Pearson. 2007. P. 705.
16. D. MOROZ. Network Features Study of the Communication Interface of Multiprocessor Modular Systems // International Scientific Journal "Computer Systems and Information Technologies". 2022. # 3. PP. 82-90.

Volodymyr KHOROSHKO **Володимир ХОРОШКО**	Dr. Sci. (Engin.), Professor of Department of Information Technology Security, National Aviation University, Kyiv, Ukraine e-mail: professor_va@ukr.net. https://orcid.org/0000-0001-6213-7086 Scopus Author ID: 7801669008	доктор технічних наук, професор, професор катедри безпеки інформаційних технологій, Національний авіаційний університет, Київ, Україна.
Serhii ZYBIN **Сергій ЗИБІН**	Dr. Sci. (Engin.), Professor of Department of Information Technology Security, National Aviation University, Kyiv, Ukraine e-mail: zysv@ukr.net. https://orcid.org/0000-0002-2670-2823 Scopus Author ID: 57202220667	доктор технічних наук, професор, професор катедри безпеки інформаційних технологій, Національний авіаційний університет, Київ, Україна.