**DOI: https://doi.org/10.15276/hait.07.2024.29**

**UDC 004.05**

# Ambiguities and their emergence conditions in self-testing of multiprocessor systems

**Vitaliy A. Romankevich**[1)]
ORCID: https://orcid.org/0000-0003-4696-5935; zavkaf@scs.kpi.ua. Scopus Author ID: 57193263058
**Kostiantyn V. Morozov**[1)]
ORCID: https://orcid.org/0000-0003-0978-6292; mcng@ukr.net. Scopus Author ID: 57222509251
**Oleksii V. Romankevich**[1)]
ORCID: https://orcid.org/0009-0006-6512-3919; alexromankevich3@gmail.com
**Lefteris Zacharioudakis**[2)]
ORCID: https://orcid.org/0000-0002-9658-3073; l.zacharioudakis@nup.ac.cy. Scopus Author ID: 57422876200
[1)] National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", 37, Peremoga Ave. Kyiv, 03056, Ukraine
[2)] Neapolis University Pafos, 2, Danais Ave. Pafos, 8042, Cyprus

## ABSTRACT

The article addresses the issue of organizing self-testing in multiprocessor systems. It examines cases where the state of certain processors (functional or faulty) remains ambiguous after executing a specific set of mutual processor tests. Determining the state of such processors requires additional capabilities, such as extra connections between processors. Ambiguity most often arises when the number of processors testing a given processor is less than the allowable number of faults. The study focuses on multiprocessor systems whose diagnostic graphs can be represented as circulant graphs, particularly graphs with two incoming and two outgoing edges. This relates to solving the problem of minimizing the number of mutual tests among system processors (each processor is tested by only two others). However, this approach can lead to ambiguity in determining the state of individual processors, especially when the allowable (and actual) number of faults in the system exceeds two. Theorems are formulated and proven to define the specific characteristics of the system for organizing mutual testing, under which the described phenomenon becomes feasible, however, in one way or another, the indices of processors with undefined states become known. The advantages of connection architectures described by circulant graphs are highlighted, particularly the fact that the number of processors in such architectures can be arbitrary – an attribute not always present in other cases (e.g., architectures with connection switches of the rectangular or hypercube type). Fault-tolerant multiprocessor systems with an allowable number of faults $T$ = 2, 3, and 4 are examined in detail. It is shown that in the case of $T$ = 2, no ambiguities arise; however, for $T$ = 4, up to three ambiguities may occur (for $T$ = 3 – up to two) depending on certain jumps in the circulant graph and specific combinations of functional and non-functional processors in the system. Examples of circulant graphs are provided where such ambiguities do not arise.

**Keywords**: Self-testing of multiprocessor systems; diagnostic graph; circulant graph; ambiguity

***For citation*: Romankevich V. A., Morozov K. V., Romankevich O. V.,Zacharioudakis L. "Ambiguities and their emergence conditions in self-testing of multiprocessor systems". *Herald of Advanced Information Technology*. 2024; Vol.7 No.4: 405–413. DOI: https://doi.org/10.15276/hait.07.2024.29**

## INTRODUCTION AND LITERATURE REVIEW

Multiprocessor systems (MS) use increases each year, for example, in control systems for complex objects such as aircraft, in systems for collecting and processing medical or banking information, and so on [1, 2], [3, 4], [5]. Modern control systems for complex objects sometimes incorporate hundreds of processors. For instance, the aircraft control systems developed by Boeing comprise 200 or more processors. The issue of organizing their self-testing is becoming increasingly important [6, 7], [8, 9], [10]. Since the failure of any processor can significantly affect the system's function, mutual during operation [11, 12], [13]. Minimizing the time required for system self-testing is one of the most critical tasks, impacting both the reliability and performance of MS [14].

In most self-testing tasks for MS, the permissible number of failures $T$ (particularly for fault-tolerant systems [15, 16], [17, 18], [19, 20], [21, 22], [23]) is predetermined. To ensure comprehensive testing, this number must derive from the number of $m$ processors testing a given processor, with the general condition $m > T$. This directly affects the number of elementary checks $N$, which, without optimization, equals $N = mn$ for $n$ processors.

The execution of self-testing processes requires certain system resources. Thus, it is clear that reducing the number of test checks is an important task in practice. For systems with a complete graph testing of processors is continuously performed topology, where any processor can test any other, the problem has been already solved, and the number of checks does not exceed $n + 2t$, where $t < n / 2$ is the actual number of faulty processors. However, such a topology of interprocessor connections is not always feasible.

Any minimization of the number of mutual tests during MS testing is inevitably associated with using only a subset of the system's interprocessor connections [24]. In particular, the minimization problem is considered for the extreme case where each processor is tested through the minimum possible 2 channels. The method's idea is to select only those connections that allow the topology to be represented as a circulant graph with 2 incoming and 2 outgoing edges.

Since the permissible number of failures $T > 2$, ambiguities can arise, i.e., situations where the state (functional or faulty) of a processor cannot be determined (this undetermined state shall be denoted as $R$). Practically, resolving the ambiguity for such a processor requires one additional check from a third, previously uninvolved processor, which necessitates the presence of a third interprocessor connection in the system. This is always feasible, as the method enables identifying the state of all processors and the $R$ processor's index.

It is known that there are topologies where such ambiguities are limited to at most one (for $T = 3$) or two (for $T = 4$), as confirmed by examples. Specifically, for $T = 3$, diagnostic graphs with circulant graph skips $S_1 = 2$ and $S_2 = 3$ are studied (where $S_i$ denotes the skips of the circulant graph), while for $T = 4$, $S_1 = 3$ and $S_2 = 4$ are used. Thus, for $T \leq 4$, the method results in no more than $2n + 2$ checks.

Circulant graphs have been extensively studied [25], and their advantages in MS have been highlighted, in particular, in the work [26]. The challenge is to explore self-testing organization for such MS to determine whether other skip values $S_i$ could lead to different number of ambiguities. This information is critical for system developers. This work addresses this question. Below, as in many works related to MS testing, it is assumed that interprocessor connections are functional.

## 1. SYSTEMS WITH 4 FAILURES

The Preparata-Metze-Chien (PMC) model [27, 28], [29, 30] is used as the fault model, as it is the most accepted in practice and wide-spread. In the PMC model, a functional processor provides a correct result (0 – functional, 1 – faulty) when testing another processor. On the other hand, if the testing processor is faulty, the result may be arbitrary, i.e. unrelated to the tested processor's state, and is denoted as $x$ on the diagnostic graph. Additionally, complete testing is only possible when $n > 2T$.

It should be noted that in such diagnostic graphs (with two incoming and two outgoing edges), ambiguities can only occur when the number of identified faulty processors – whose states are resolved through the execution of all possible tests and analysis of the results – is less than $T$. Additionally, a processor must be tested either by two faulty processors or by one faulty processor and one in state $R$ (although in this latter case, the ambiguity may be avoided).

***Theorem 1.*** For $T = 4$, there exist multiprocessor systems (MS) with diagnostic graphs of the considered type, where, under certain arrangements of functional and faulty processors, three ambiguities may remain after performing all possible tests. For this, the condition $3S_1 = 3S_2 \bmod n$ is sufficient.

The proof is based on the following idea: for three ambiguities to exist, it is sufficient that three faulty processors form three pairs, each pair testing the same processor $R_1$, $R_2$, or $R_3$. A fourth faulty processor (if present in the system under investigation) is among $R_i$ $(i = 1, 2, 3)$. Thus, determining the exact state of all these processors is indeed impossible without additional tests involving other processors.

To illustrate such an «impossibility» let us consider a simple example of a diagnostic graph with nine vertices and skips $S_1 = 1$ and $S_2 = 4$ (Fig. 1).

Here, $F$ represents a faulty processor, $G$ a functional one, and the arrows indicate test results. Processor numbers correspond to graph vertices. To the left of the graph, the 0-chain is depicted, and all 1-results are listed. To the right, the states of all processors are determined based on the analysis performed using the algorithm from, grounded in the following *Proposition*, which follows directly from the PMC model and method.

Let there be a set of processors $\alpha_i$ connected to processor $A_i$ by zero-result edges. Let the weight $P_i$ of processor $A_i$ is the cardinality of the set $\alpha_i$ (including $A_i$ itself). Let $b_i$ be the number of disjoint processor pairs (with one-result edges) that are not in $\alpha_i$. Then $A_i$ is functional if $P_i + b_i > T - t$ holds, where $t$ is the number of faulty processors identified so far (the analysis is conducted sequentially over time).

This proposition allows for identifying functional processors. Faulty processors are determined via testing by functional ones.

For example, in Fig. 1, the diagram of the 0-chain allows us to determine that the weight of processor 9 is 3. Additionally, there are two pairs of processors with one-result checks: (6, 1) and (3, 4), which are not part of the set {5, 8, 9}. According to the stated proposition, we can conclude that processor 9 is functional. This processor tests processors 1 and 4, both of which produce unit results, indicating their malfunction according to the PMC model. Next, analyzing the 0-chain diagram, we determine the weight of processor 3, which equals 2. According to the rule, processor 3 is operational because $t = 2$ and there is a pair (6, 7). Processor 3 tests processor 7 with a one-result, indicating that 7 is faulty, and $t = 3$. Since the weight of processor 6 is also 2 according to the same 0-chain diagram, we can conclude, based on the proposition, that it is functional. Further analysis does not reveal any new information, meaning the status of processors 2, 5, and 8 remains undefined.

It is of interest to define the conditions (essentially the values of $S_i$ and $n$) under which these 3 ambiguities occur in the general case. According to the above «idea», assuming $S_1 < S_2 < n$ for simplicity, the following can be expressed:

$$F_1 + S_1 = R_1 \bmod n;$$
$$F_2 + S_1 = R_2 \bmod n;$$
$$F_3 + S_1 = R_3 \bmod n;$$
$$F_3 + S_2 = R_1 \bmod n;$$
$$F_1 + S_2 = R_2 \bmod n;$$
$$F_2 + S_2 = R_3 \bmod n.$$

By performing simple transformations, the condition for the occurrence of three ambiguities can be obtained:

$$3S_1 = 3S_2 \bmod n.$$

In this case, the arrangement of functional and faulty processors must correspond to the «idea» described above. The theorem is proven.

Based on the obtained relationship, a table (Table 1) can be compiled for certain values of $S$ and $n$ for $T = 4$, where three ambiguities may arise. Using this table, a developer can easily calculate values that match the parameters of their multiprocessor system. When compiling the table, certain recommendations were taken into account, namely, $S_2$ must not be a multiple of $S_1$ (otherwise, the graph may split into disconnected subgraphs). This condition applies to cases where $S_1 > 1$.

The obtained results (including Table 1) indicate that, for example, the values of $S_1 = 2$ and $S_2 = 3$ do not lead to the appearance of three ambiguities for any value of $n$. In this case, the total number of checks may increase by one but does not exceed $2n + 3$.

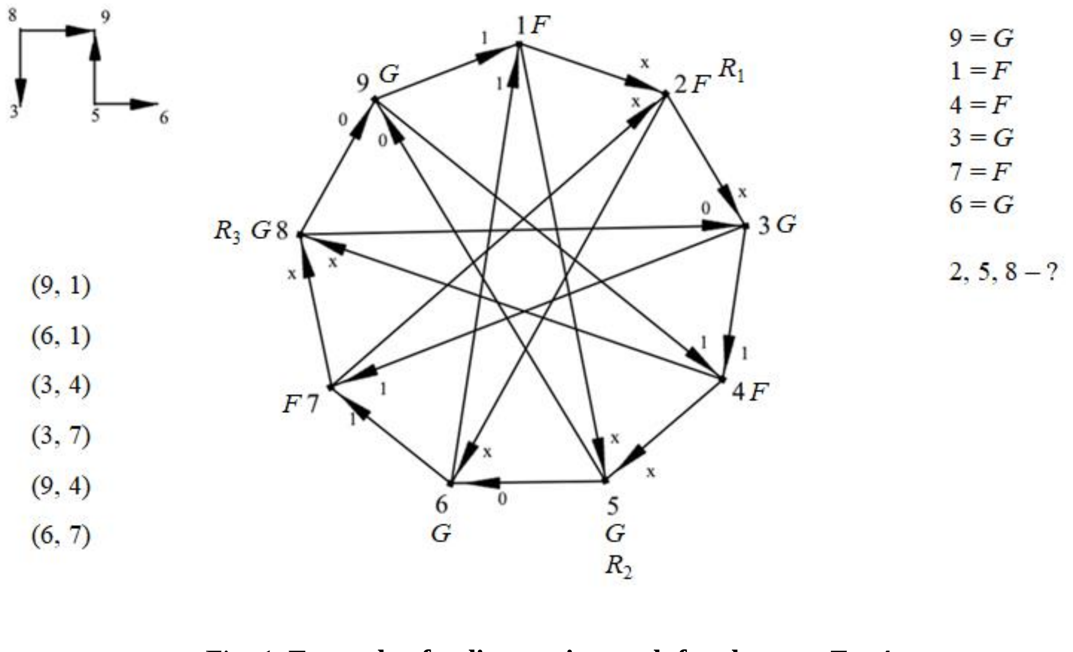

*Fig. 1.* **Example of a diagnostic graph for the case $T = 4$**
***Source:*** **compiled by the authors**

*Table 1.* **Table of MS parameter values for *T* = 4**

| $S_1$ | $S_2$ | $n$ | $S_1$ | $S_2$ | $n$ | $S_1$ | $S_2$ | $n$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 4 | 9 | 2 | 5 | 9 | 3 | 7 | 12 |
| 1 | 5 | 12 | 2 | 7 | 15 | 3 | 8 | 15 |
| 1 | 6 | 15 | 2 | 9 | 21 | 3 | 10 | 21 |
| 1 | 7 | 18, 9 | 2 | 11 | 27 | 3 | 11 | 24, 12 |

***Source:*** **compiled by the authors**

*Table 2.* **Table of MS parameter values for *T* = 3**

| $S_1$ | $S_2$ | $n$ | $S_1$ | $S_2$ | $n$ | $S_1$ | $S_2$ | $n$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 5 | 8 | - | - | - | 3 | 7 | 8 |
| 1 | 6 | 10 | 2 | 7 | 10 | 3 | 8 | 10 |
| 1 | 7 | 12 | 2 | 9 | 14 | 3 | 10 | 14 |
| 1 | 8 | 14 | 2 | 11 | 18 | 3 | 11 | 16 |

***Source:*** **compiled by the authors**

## 2. SYSTEMS WITH 3 FAILURES

The previously stated «idea» transforms as follows for the case $T = 3$: two out of three faulty processors test the same pair of processors, while the third faulty processor is either among the tested ones (and remains undetected) or absent. This essentially validates the following theorem.

***Theorem 2.*** For $T = 3$, there exist MS with diagnostic graphs of the considered type, where, under specific arrangements of functional and faulty processors, two ambiguities may remain after performing all possible tests. For this, the condition $2S_1 = 2S_2 \bmod n$ is sufficient.

An example in Fig. 2 illustrates this possibility.

Similarly to the case $T = 4$, we can write:

$$H_1 + S_1 = H_2 + S_2 \bmod n;$$
$$H_2 + S_1 = H_1 + S_2 \bmod n.$$

This leads to the condition:

$$2S_1 = 2S_2 \bmod n.$$

Some values of $S$ and $n$ for $T = 3$, where two ambiguities may occur, are listed in Table 2.

It should be noted that in the graph (Fig. 2), there are two vertices (5th and 10th), corresponding to processors that are tested by one faulty processor and one *R*-type processor, and their states are determined. At the same time, it may seem that if the third faulty processor is hidden among the *R*-type processors, its state also cannot be determined, resulting in the appearance of a third *R*. However, this is not the case; moreover, it may turn out that ambiguities disappear.

Let us illustrate this with an example: if processor $F_3$ is assigned index 5, then the test $8 \rightarrow 5$ will yield a new 1 (recall that by this point, two faulty processors – 1 and 6 – have already been identified). Consequently, processor 3 is functional: its weight equals 1, and among the two others, not connected to it in any way (processors 5 and 8), at least one is faulty. For the case $T = 4$, it can be said that a fourth ambiguity will not arise in similar situations.

The algorithm for performing mutual checks during self-testing consists of the following steps:

- perform all possible test checks and construct a diagnostic graph;
- build a 0-chain;

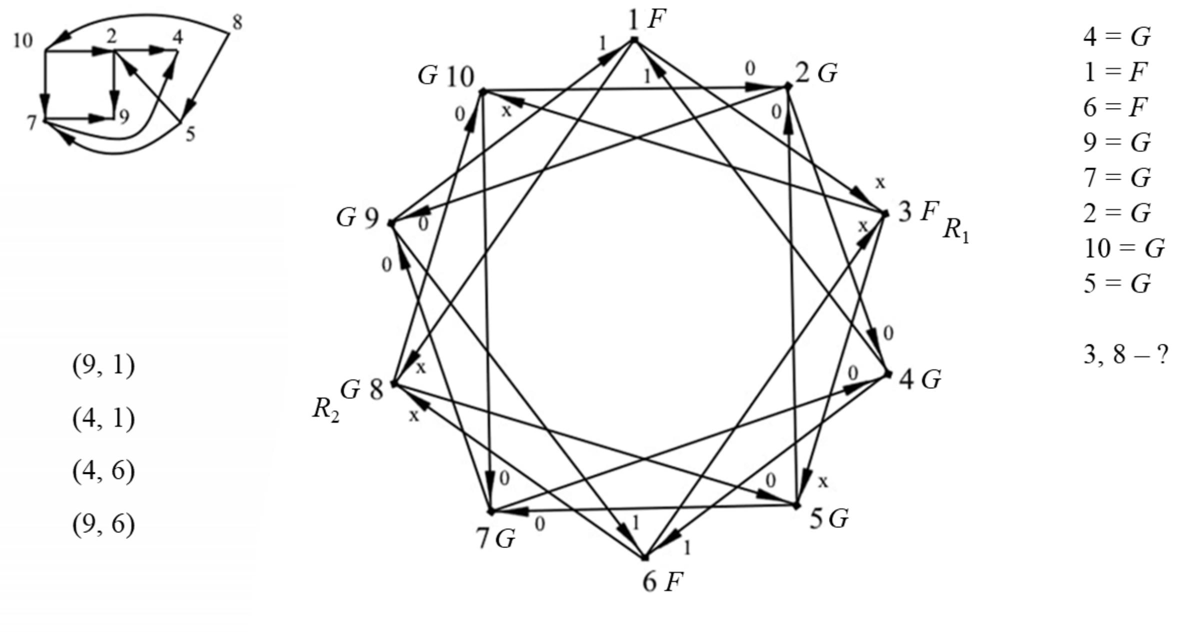

*Fig. 2.* **Example of a diagnostic graph for the case *T* = 3**
***Source:*** **compiled by the authors**

- list all pairs of vertices with 1-checks;
- using the aforementioned *Proposition*, determine the states of processors for which it is possible;
- treat the remaining processors as having indeterminate states, for which additional checks using other communication channels are required.

It can be noted that, broadly speaking, a third ambiguity can arise only at vertices marked as *xx*. In practice, during testing, the *x* value is also clarified, but the overall result of the analysis does not change.

It should be noted that the obtained result can be generalized for arbitrary values of *T*.

***Theorem 3.*** For any *T*, there exist MS and their diagnostic graphs of the considered type, where after performing all possible tests, $T-1$ ambiguities may remain.

Indeed, $T-1$ vertices corresponding to $T-1$ faulty processors (one of the faulty processors remains undetected) can be arranged within the graph in such a way that they form $T-1$ pairs. Each pair corresponds to a pair of processors testing the same processor, whose state cannot be determined without using additional capabilities.

Let us briefly discuss the efficiency of the proposed approach. Several methods are known for organizing self-testing and determining the state of processors in multiprocessor systems. These methods take into account the specifics of fault models (Preparata-Metze-Chein, Barzi-Grandoni, intermittent faults, etc.) on the one hand, and on the other, the features of the interconnection architectures between processors (matrix architecture, hypercube, circulants, etc.). The most complex architecture is, undoubtedly, one with arbitrarily chosen interconnections, which is most often encountered in control systems for complex objects (computing systems are usually homogeneous).

In practice, developers most often use the well-known majority method (it can be said to belong to von Neumann). In this approach, each processor is tested by an odd number of other processors, and the majority of the decisions determines the correctness of the tested processor. If the system is tolerant to a single fault, each processor must be tested by three others. In the general case, when the system is tolerant to *T* faults, at least $2T+1$ connections are required to test each processor, and the total number of checks, $N = n(2T+1)$. For $T = 4$, as in the example above, $N = 9n$. At the same time, the proposed method results in $2n+3$ checks, which is significantly fewer.

Considering the fact that self-testing is performed continuously in control systems, reducing the time required for self-testing improves the system's performance. This is in addition to enhancing reliability by minimizing the duration of incorrect system operation upon the occurrence of each new failure.

## CONCLUSIONS

This study investigates the peculiarities of minimizing the number of mutual processor checks in a multiprocessor system, where the diagnostic graph is a subset of the system's connection topology, specifically a circulant graph with two incoming and two outgoing edges per vertice. Various combinations of skip values in this graph are analyzed, showing that under certain arrangements of faulty and functional processors, the number of processors with undetermined states may exceed previously anticipated levels. Conditions for such situations are provided.

It is noted that system developers, aware of these scenarios, should allocate the necessary connections in the system to perform the required additional tests. In general, no more than $2n+T-1$ tests are needed (for $T = 2$, no ambiguities arise, requiring at most $2n$ tests).

As a remark, the authors do not consider degenerate cases of circulants that developers would not select for testing. These include circulants with skips *i* and $n-i$, cases where one skip equals half of *n*, and others. Hence, the theorems use the term “exist”.

## REFERENCES

1. Nazarova, O. S., Osadchyy, V. V. & Rudim, B. Y. “Computer simulation of the microprocessor liquid level automatic control system”. *Applied Aspects of Information Technology.* 2023; 6 (2): 163–174. DOI: https://doi.org/10.15276/aait.06.2023.12.
2. Kotov, D. O. “A generalized model of an adaptive information-control system of a car with multi-sensor channels of information interaction”. *Applied Aspects of Information Technology*. 2021; 5 (1): 25–34. DOI: https://doi.org/10.15276/aait.05.2022.2.
3. Antoniuk, V. V., Drozd, M. O. & Drozd, O. B. “Power-oriented checkability and monitoring of the current consumption in FPGA projects of the critical applications”. *Applied Aspects of Information Technology*. 2019; 2 (2): 105–114. DOI: https://doi.org/10.15276/aait.02.2019.2.

4. Kovalev, I. S., Drozd, O. V., Rucinski, A., Drozd, M. O., Antoniuk, V. V. & Sulima, Y. Y. “Development of computer system components in critical applications: problems, their origins and solutions”. *Herald of Advanced Information Technology.* 2020; 3 (4): 252−262. DOI: https://doi.org/10.15276/hait.04.2020.4.

5. Fang, Y., Lu, Y., Han, T., Hu, Q. “Design of missile incremental adaptive fault tolerant control system”. *Beijing Hangkong Hangtian Daxue Xuebao/Journal of Beijing University of Aeronautics and Astronautics.* 2022; 48 (5): 920−928, https://www.scopus.com/record/display.uri?eid=2-s2.0-85130918044&origin=reflist. DOI: https://doi.org/10.13700/j.bh.1001-5965.2021.0454.

6. Duarte, E. P., Rodrigues, L. A., Camargo, E. T. & Turchetti, R. C. “The missing piece: a distributed system-level diagnosis model for the implementation of unreliable failure detectors”. *Computing.* 2023. 105 (12): 2821−2845, https://www.scopus.com/record/display.uri?eid=2-s2.0-85168337693&origin=resultslist. DOI: https://doi.org/10.1007/s00607-023-01211-8.

7. Shtanenko, S., Samokhvalov, Y., Toliupa, S. & Silko, O. “The approach to assessment of technical condition of microprocessor systems that are implemented on integrated circuits with a programmable structure“. *Lecture Notes in Electrical Engineering.* 2023; 965 (LNEE): 495−508, https://www.scopus.com/record/display.uri?eid=2-s2.0-85151045762&origin=resultslist.
DOI: https://doi.org/10.1007/978-3-031-24963-1_28.

8. Mánik, M. & Gramatová, E. “Efficient diagnostics algorithms for regular computing structures”. *Proceedings of the 14th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems*. 2011. p. 87−92, https://www.scopus.com/record/display.uri?eid=2-s2.0-79959980500&origin=resultslist. DOI: https://doi.org/10.1109/DDECS.2011.5783054.

9. Drozd, J., Drozd, A. & Al-Dhabi, M. “A resource approach to on-line testing of computing circuits”. *Proceedings of 2015 IEEE East-West Design and Test Symposium.* 2015. p. 276−281, https://www.scopus.com/record/display.uri?eid=2-s2.0-84979266558&origin=resultslist.
DOI: https://doi.org/10.1109/EWDTS.2015.7493122.

10. Bagchi, A., Servatius, B. & Shi, W. “2-Satisfiability and diagnosing faulty processors in massively parallel computing systems”. *Discrete Applied Mathematics.* 1995; 60 (1-3): 25−37, https://www.scopus.com/record/display.uri?eid=2-s2.0-58149208793&origin=resultslist.
DOI: https://doi.org/10.1016/0166-218X(94)00041-B.

11. Drozd, A., Drozd, J., Antoshchuk, S., Nikul, V. & Al-Dhabi, M. “Objects and methods of on-line testing: main requirements and perspectives of development”. *Proc. IEEE East-West Design & Test Symp*. Yerevan, Armenia. 2016. p. 72–76, https://www.scopus.com/record/display.uri?eid=2-s2.0-85015150287&origin=resultslist. DOI: https://doi.org/10.1109/EWDTS.2016.7807750.

12. Lu, Q., Gui, W. & Su, M. “A fireworks algorithm for the system-level fault diagnosis based on MM* model”. *IEEE Access.* 2019; 7: 136975−136985, https://www.scopus.com/record/display.uri?eid=2-s2.0-85077808842&origin=resultslist. DOI: https://doi.org/10.1109/ACCESS.2019.2942336.

13. Hakimi, S.L. & Amin, A.T. “Characterization of connection assignment of diagnosable systems”. *IEEE Transactions on Computers*: 1974; C-23 (1): 86−88, https://www.scopus.com/record/display.uri?eid=2-s2.0-84947657515&origin=resultslist. DOI: https://doi.org/10.1109/T-C.1974.223782.

14. Wang, Z. et al. “Research on joint optimal scheduling of task energy efficiency and reliability in heterogeneous multiprocessor real-time system”. *IEEE 2nd International Conference on Power, Electronics and Computer Applications (ICPECA)*. 2022. p. 17−22, https://www.scopus.com/record/display.uri?eid=2-s2.0-85127423511&origin=resultslist. DOI: https://doi.org/10.1109/ICPECA53709.2022.9719271.

15. Nedeljkovic, J. N., Dosic, S. M. & Nikolic, G. S. “A survey of hardware fault tolerance techniques”. *2023 58th International Scientific Conference on Information, Communication and Energy Systems and Technologies, ICEST. Proceedings*. 2023. p. 223−226, https://www.scopus.com/record/display.uri?eid=2-s2.0-85167870342&origin=resultslist. DOI: https://doi.org/10.1109/ICEST58410.2023.10187275.

16. Godabole, P. & Bhole, G. “Utilization and criticality based fault-tolerant scheduling in multicore mixed critical systems”. *International Journal of Pervasive Computing and Communications.* 2024; 20 (1): 126−146, https://www.scopus.com/record/display.uri?eid=2-s2.0-85149644279&origin=resultslist.
DOI: https://doi.org/10.1108/IJPCC-06-2022-0248.

17. Safari, S. et al. “A survey of fault-tolerance techniques for embedded systems from the perspective

of power, energy, and thermal issues". *IEEE Access*. 2022: (10): 12229−12251, https://www.scopus.com/record/display.uri?eid=2-s2.0-85123365684&origin=resultslist.
DOI: https://doi.org/10.1109/ACCESS.2022.3144217.

18. Abbaspour, A., Mokhtari, S., Sargolzaei, A. & Yen, K. K. "A survey on active fault-tolerant control systems". *Electronics*. 2020; 9 (9): 1−23, https://www.scopus.com/record/display.uri?eid=2-s2.0-85090902832&origin=resultslist. DOI: https://doi.org/10.3390/electronics9091513.

19. Hu, Q., Niu, G. & Wang, C. "Spacecraft attitude fault-tolerant control based on iterative learning observer and control allocation". *Control Allocation for Spacecraft under Actuator Faults*. 2018; 75: 245−253, https://www.scopus.com/record/display.uri?eid=2-s2.0-85041462830&origin=resultslist.
DOI: https://doi.org/10.1016/j.ast.2017.12.031.

20. Joshi, H. & Sinha, N. K. "Adaptive fault tolerant control design for stratospheric airship with actuator faults". *IFAC-PapersOnLine*. 2022; 55 (1): 819−825, https://www.scopus.com/record/display.uri?eid=2-s2.0-85132157930&origin=resultslist. DOI: https://doi.org/10.1016/j.ifacol.2022.04.134.

21. Dumitrescu, M. "Fault tolerant control multiprocessor systems modeling using advanced stochastic petri nets". *Procedia Technology*. 2016; 22: 623−628. DOI: https://doi.org/10.1016/j.protcy.2016.01.129.

22. Nair, P.P., Sarkar, A. & Biswas, S. "Fault-tolerant real-time fair scheduling on multiprocessor systems with cold-standby". *IEEE Transactions on Dependable and Secure Computing.* 2021; 18 (4): 1718-1732, https://www.scopus.com/record/display.uri?eid=2-s2.0-85070947169&origin=reflist.
DOI: https://doi.org/10.1109/TDSC.2019.2934098.

23. Pathan, R. M. "Fault-tolerant and real-time scheduling for mixed-criticality systems". *Real-Time Systems.* 2014; 50 (4): 509−547, https://www.scopus.com/record/display.uri?eid=2-s2.0-84903574751&origin=reflist. DOI: https://doi.org/10.1007/s11241-014-9202-z.

24. Romankevich, A. M., Morozov, K. V. & Romankevich V A. "Generalization of the formal method for determining the state of processors of a multiprocessor system under testing". *Lecture Notes on Data Engineering and Communications Technologies.* 2022; 134: 363−375, https://www.scopus.com/record/display.uri?eid=2-s2.0-85129575140&origin=resultslist.
DOI: https://doi.org/10.1007/978-3-031-04812-8_31.

25. Mithun, B., Laxman, S., Gour, K. D. & Kalishankar, T. "Fault-tolerant metric dimension of circulant graphs Cn(1,2,3)". *Theoretical Computer Science*. 2020; 817: 66−79, https://www.scopus.com/record/display.uri?eid=2-s2.0-85060525332&origin=resultslist.
DOI: https://doi.org/10.1016/j.tcs.2019.01.011.

26. Romankevich, V. A., Romankevich, A. V. & Akhmedova, D. N. "Method of decreasing number of mutual testing during self-testing of multiprocessor systems" (In Russian). *Radioelectronic and Computer Systems.* 2018; 4: 61−66. DOI: https://doi.org/10.32620/reks.2018.4.06.

27. Preparata, F. P., Metze, G. & Chien, R. T. "On the connection assignment problem of diagnosable systems". *IEEE Transactions on Electronic Computers*. 1967; ES-16 (6): 848−854, https://www.scopus.com/record/display.uri?eid=2-s2.0-84938017623&origin=resultslist.
DOI: https://doi.org/10.1109/PGEC.1967.264748.

28. Mánik, M. & Gramatová, E. "Diagnosis of faulty units in regular graphs under the PMC model". *Proceedings of the 2009 12th International Symposium on Design and Diagnostics of Electronic Circuits & Systems.* 2009, p. 202−205, https://www.scopus.com/record/display.uri?eid=2-s2.0-70349337043&origin=resultslist. DOI: https://doi.org/10.1109/DDECS.2009.5012128.

29. Chang, N.-W. & Hsieh, S.-Y. "Conditional diagnosability of alternating group networks under the PMC model". *IEEE/ACM Transactions on Networking.* 2020; 28 (5): 1968−1980, https://www.scopus.com/record/display.uri?eid=2-s2.0-85105805249&origin=resultslist.
DOI: https://doi.org/10.1109/TNET.2020.3002093.

30. Liu, J., Zhou, Q., Yu, Zh. & Zhou, Sh. "Fault diagnosability of regular networks under the hybrid pmc model". *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics).* 2021; 13025 (LNCS): 283−297, https://www.scopus.com/record/display.uri?eid=2-s2.0-85118183684&origin=resultslist.
DOI: https://doi.org/10.1007/978-3-030-89543-3_24.

**Conflicts of Interest:** The authors declare that they have no conflict of interest regarding this study, including financial, personal, authorship or other, which could influence the research and its results presented in this article

Received    06.09.2024
Received after revision    15.11.2024
Accepted    21.11.2024

**DOI: https://doi.org/10.15276/hait.07.2024.29**
**УДК 004.05**

# Невизначеності та умови їх виникнення при самотестуванні багатопроцесорних систем

**Романкевич Віталій Олексійович**[1)]
ORCID: https://orcid.org/0000-0003-4696-5935; zavkaf@scs.kpi.ua. Scopus Author ID: 57193263058
**Морозов Костянтин В’ячеславович**[1)]
ORCID: https://orcid.org/0000-0003-0978-6292; mcng@ukr.net. Scopus Author ID: 57222509251
**Романкевич Олексій Віталійович**[1)]
ORCID: https://orcid.org/0009-0006-6512-3919; alexromankevich3@gmail.com
**Лефтеріс Захаріудакіс**[2)]
ORCID: https://orcid.org/0000-0002-9658-3073; l.zacharioudakis@nup.ac.cy. Scopus Author ID: 57422876200
1) Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», пр. Перемоги, 37. Київ, 03056, Україна
[2)] Університет Неаполіс Пафос, Данаіс Авеню. 2. Пафос, 8042, Кіпр

## АНОТАЦІЯ

Стаття присвячена проблемі організації самотестування багатопроцесорних систем. Аналізуються випадки, коли стан деяких процесорів (справний чи несправний) залишається невизначеним після виконання певної множини взаємних тестувань процесорів. Для встановлення стану таких процесорів потрібне використання деяких додаткових можливостей, наприклад додаткових зв’язків між процесорами. Невизначеність виникає частіше за все у таких випадках, коли кількість процесорів, які тестують даний процесор, є меншою за допустиму кількість відмов. Досліджуються багатопроцесорні системи, діагностичні графи яких можна представити графами-циркулянтами, зокрема графами з двома вхідними та двома вихідними дугами. Це пов’язано з вирішенням задачі мінімізації кількості взаємоперевірок процесорів системи (дійсно, кожний процесор тестується всього лише двома іншими). Проте при цьому може виникнути певна невизначеність у процесі встановлення стану окремих процесорів, і це може бути саме тоді, коли кількість допустимих (а також наявних) відмов у системі перевищує 2. Формулюються та доводяться теореми, які визначають конкретні характеристики системи організації взаємотестувань, коли описане явище стає можливим, але так чи інакше номери процесорів, стан яких не визначено, стають відомими. Відзначаються переваги архітектур зв’язків, котрі можуть бути описані графами-циркулянтами, зокрема те, що кількість процесорів в них може бути довільною, що не завжди має місце в інших випадках (наприклад в архітектурах з комутаторами зв’язків типу прямокутник або гіперкуб). Детально розглядаються відмовостійкі багатопроцесорні системи з допустимим числом відмов $T = 2$, 3 та 4. Показується, що у випадку $T = 2$ невизначеності не виникають, але при $T = 4$ їх може виникнути до трьох (у випадку $T = 3$ – до двох) при деяких стрибках у графі-циркулянті та деяких комбінаціях розміщення справних та несправних процесорів у системі. Наводяться приклади графів-циркулянтів, коли подібне не має місця.

**Ключові слова*:*** самотестування багатопроцесорних систем; діагностичний граф; граф-циркулянт; невизначеність

## ABOUT THE AUTHORS

**Vitaliy A. Romankevich** - Doctor of Engineering Sciences, Professor, Head of System Programming and Specialized Computer System Department. National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”, 37, Peremoga Ave. Kyiv, 03056, Ukraine
ORCID: https://orcid.org/0000-0003-4696-5935; zavkaf@scs.kpi.ua. Scopus Author ID: 57193263058
***Research field***: Fault-tolerant multiprocessor systems reliability estimation, GL-models; Self-diagnosable systems; diagnosis of multiprocessor systems; discrete mathematics

**Романкевич Віталій Олексійович** - доктор технічних наук, професор, завідувач кафедри Системного програмування і спеціалізованих комп’ютерних систем, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», пр. Перемоги, 37.  Київ, 03056, Україна

**Kostiantyn V. Morozov** - PhD (Eng), Assistant, System Programming and Specialized Computer System Department. National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”, 37, Peremoga Ave. Kyiv, 03056, Ukraine
ORCID: https://orcid.org/0000-0003-0978-6292, mcng@ukr.net, Scopus Author ID: 57222509251
***Research field*:** GL-models, Fault-tolerant multiprocessor systems reliability estimation, Self-diagnosable multiprocessor systems.

**Морозов Костянтин В’ячеславович** - кандидат технічних наук, асистент кафедри Системного програмування і спеціалізованих комп’ютерних систем, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», пр. Перемоги, 37. Київ, 03056, Україна

**Oleksii V. Romankevich** - Master's student, System Programming and Specialized Computer System Department. National Technical University of Ukraine “Igor Sikorsky Kyiv Polytechnic Institute”, 37, Peremoga Ave. Kyiv, 03056, Ukraine
ORCID: https://orcid.org/0009-0006-6512-3919, alexromankevich3@gmail.com
***Research field*:** Self-diagnosable systems; diagnosis of multiprocessor systems

**Романкевич Олексій Віталійович** - магістрант кафедри Системного програмування і спеціалізованих комп’ютерних систем, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», пр. Перемоги, 37. Київ, 03056, Україна

**Lefteris Zacharioudakis** - PhD (Eng), Assistant professor, Neapolis University Pafos, 2, Danais Ave. Pafos, 8042, Cyprus
ORCID: https://orcid.org/0000-0002-9658-3073; l.zacharioudakis@nup.ac.cy. Scopus Author ID: 57422876200
***Research field*:** Principles of cybersecurity; operating systems; information security; cryptography; penetration testing

**Лефтеріс Захаріудакіс –** кандидат технічних наук, доцент, Університет Неаполіс Пафос, Данаіс Авеню, 2. Пафос, 8042, Кіпр.