UDC 004.052.3:004.22
DOI: 10.25140/2411-5363-2019-1(15)-134-144

Heorhii Loutskii, Artem Volokyta, Pavlo Rehida, Olexandr Goncharenko

USING EXCESS CODE TO DESIGN FAULT-TOLERANT TOPOLOGIES

Urgency of the research. *The task of increasing fault tolerance is one of the key tasks in constructing a computing system and in modernizing an already existing one. Particular attention is paid to it when building multicomputer systems or clusters. The most interesting ways to increase fault tolerance is to use the topological structure of the system to bypass the malfunction and use one or another element of the system to replace the faulty. Of course, this requires the development of a specific topology. The article deals with the development of fault-tolerant versions of popular topologies, such as quasi-quantum and hypercube, based on the excess code 0/1 /-1.*

Target setting. *An important part of any multicomputer system or cluster is its topological structure. This structure defines the routing of messages in the system, speed of message transmission, and level of fault-tolerance of a system. The article proposes a method for increasing fault-tolerance based on the use of excess code.*

Actual scientific researches and issues analysis. *Synthesis of topologies such as the hypercube or the de Bruin topology is well studied and described now, there are papers consider methods for increasing the fault-tolerance that based on usage of additional nodes that duplicate current nodes. Other papers consider using a tree-based routing to improve fault-tolerance of the system.*

Uninvestigated parts of general matters defining. *Now the possibilities of using excess code 0/1/-1 for creating new fault-tolerant topologies based on existing synthesis methods are unconsidered.*

The research objective. *The task is to describe the synthesis of fault tolerant topologies, the consideration of the possibilities of using their features and the analysis of the main characteristics in comparison with each other and with classic versions based on binary code.*

The statement of basic materials. *The synthesis of hypercube and de Bruin topology is described on the basis of the usual binary code and the redundant code 0/1 / -1, the possibilities of using redundancy are considered, first of all, to increase the fault-tolerance, a comparative analysis of all of these topologies is carried out.*

Conclusions. *The analysis of characteristics is performed, the main advantages and disadvantages of the proposed topological structures are highlighted, suggestions for their improvement are made.*

Keywords: *fault tolerance; de Bruin; hypercube; excess code.*

Fig.: 13. Table: 7. References: 8.

Target setting. An important part of any multicomputer system or cluster is its topological structure. It describes how computing elements are interconnected. This structure defines the routing of messages in the system, speed of message transmission, and level of fault-tolerance of a system. Level of fault-tolerance represents the ability of the system to function correctly in the failure of its constituent parts. Therefore, we propose a method to improve fault tolerance, based on the usage of excess code.

Actual scientific researches and issues analysis. Synthesis of topologies such as the hypercube or the de Bruin topology (also called quasi-quantum) is well studied and described. Many papers consider methods for increasing the fault-tolerance that based on usage of additional nodes that duplicate current nodes. Other papers consider using a tree-based routing to improve fault-tolerance of the system and to prevent locks.

Uninvestigated parts of general matters defining. Now the possibilities of using excess code 0/1/-1 for creating new fault-tolerant topologies based on existing synthesis methods are unconsidered.

The research objective. The purpose of the research is a synthesis of new topologies based on quasi-quantum and hypercube topologies using an excess code for encoding nodes numbers. Also, it is necessary to make a detailed analysis of the system's characteristics and routing capabilities. Special attention will be paid to the issues of fault-tolerance.

The statement of basic materials. The task of topology synthesis defined: there are M nodes, their numbers are encoded by some code. Usually, $M = B^N$, where B – the basis of the chosen numeral system or code, N – a numeral bit capacity of nodes numbers. One of the goals of this task to get for each node all codes of its neighbors. To achieve that some transformation with node's code is used.

Basic definitions. De Bruin topology and the hypercube are quite convenient in terms of routing. In the classic version [1-4], the binary numeral system is used for encoding of nodes numbers. It means, the maximal count of nodes for N – bit code is 2^N. The difference between

these topologies is as follows: to synthesis de Bruin topology Shuffle transformation is used, and to synthesis hypercube Exchange is used. In the binary numeral system, these approaches looks like shift and bit's inversion respectively.

Synthesis of topology using binary code. Below, the synthesis process for each of the selected topologies in the case of N=2 and N=3 is shown.

In de Bruin topology getting of neighbor node's code performs through shifts on left and right with insertion 0 or 1 in released bit. Table 1 shows synthesis of this topology for case N = 2, table 2 – for N = 3.

Table 1

De Bruin topology synthesis for 2-bits binary code

Value	Code	Put 0 to released bit		Put 1 to released bit	
		Shift left	Shift right	Shift left	Shift right
0	00	00	00	01	10
1	01	10	00	11	10
2	10	00	01	01	11
3	11	10	01	11	11

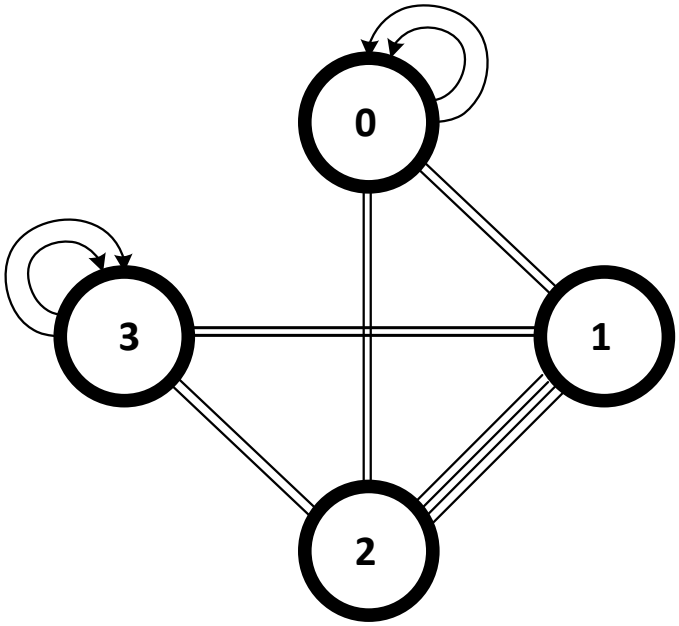

Fig. 1. De Bruin topology for N=2

Table 2

De Bruin topology synthesis for 3-bits binary code

Value	Code	Put 0 to released bit		Put 1 to released bit	
		Shift left	Shift right	Shift left	Shift right
0	000	000	000	001	100
1	001	010	000	011	100
2	010	100	001	101	101
3	011	110	001	111	101
4	100	000	010	001	110
5	101	010	010	011	110
6	110	100	011	101	111
7	111	110	011	111	111

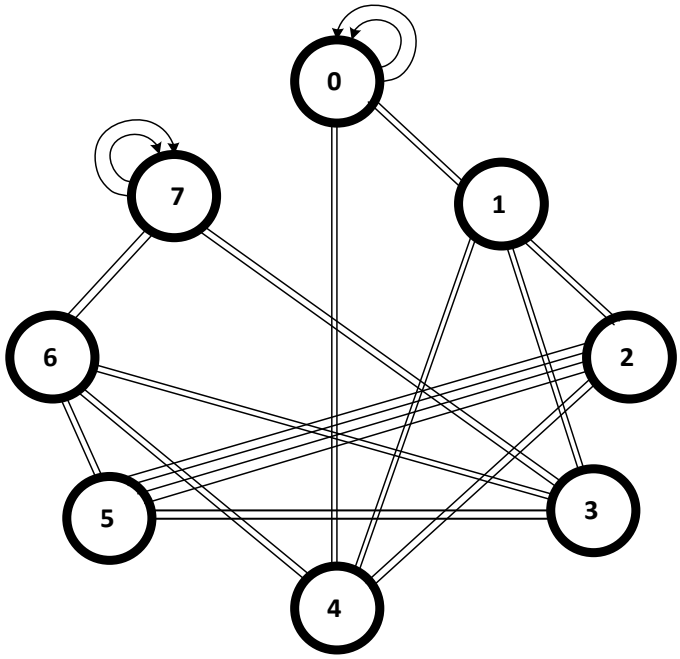

Fig. 2. De Bruin topology for N = 3

Analogically we can use similar for hypercube synthesis [5] but it is required to do inversions of one digit of code instead shift with insertion. It is illustrated in tables 3 and 4.

Table 3

Hypercube topology synthesis for N=2

Value	**Code**	Exchange 1st bit	Exchange 2nd bit
0	00	10	01
1	01	11	00
2	10	00	11
3	11	01	10

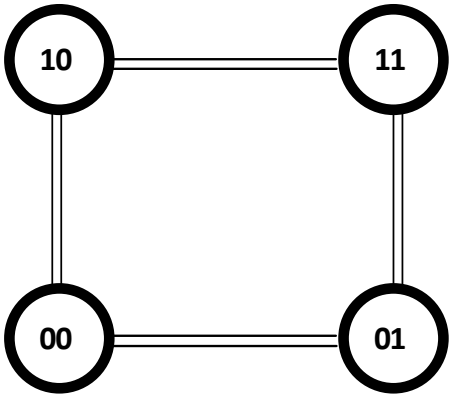

Fig. 3. Hypercube, N = 2

Table 4

Hypercube topology synthesis for N=3

Value	**Code**	Exchange 1st bit	Exchange 2nd bit	Exchange 3rd bit
0	000	100	010	001
1	001	101	011	000
2	010	010	000	011
3	011	011	001	010
4	100	000	110	101
5	101	001	111	100
6	110	010	100	111
7	111	011	101	110

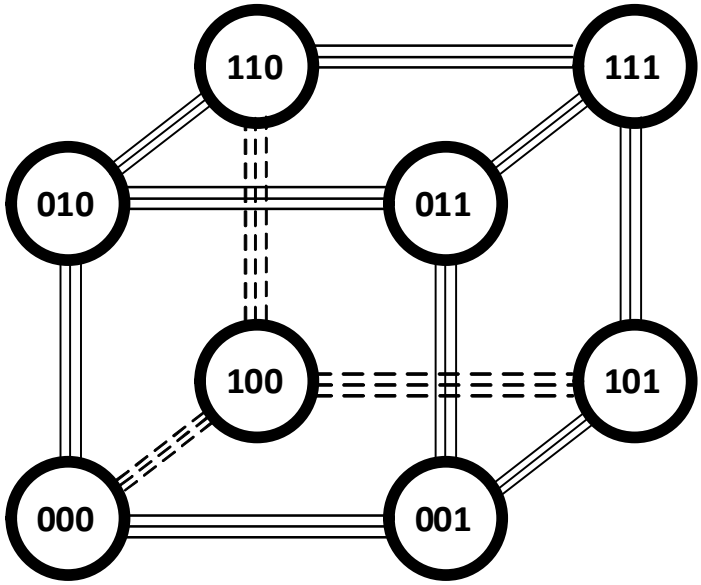

Fig. 4. Hypercube, N = 3

About excess code 0/1/-1. The excess code 0/1/-1 differs from the usual binary code with the presence of an additional number -1. It is denoted by the character T. All other properties, including the weight of the digits, are the same as for usual binary code. The main feature of this code is: one number in this code sometimes can be presented in several ways. For example, the number 3 in 3-digits excess code can be written as 011, 10T and 1T1. The maximal count of representations growth with a bit capacity of code. Also, unlike to usual binary code, this code includes the negative numbers.

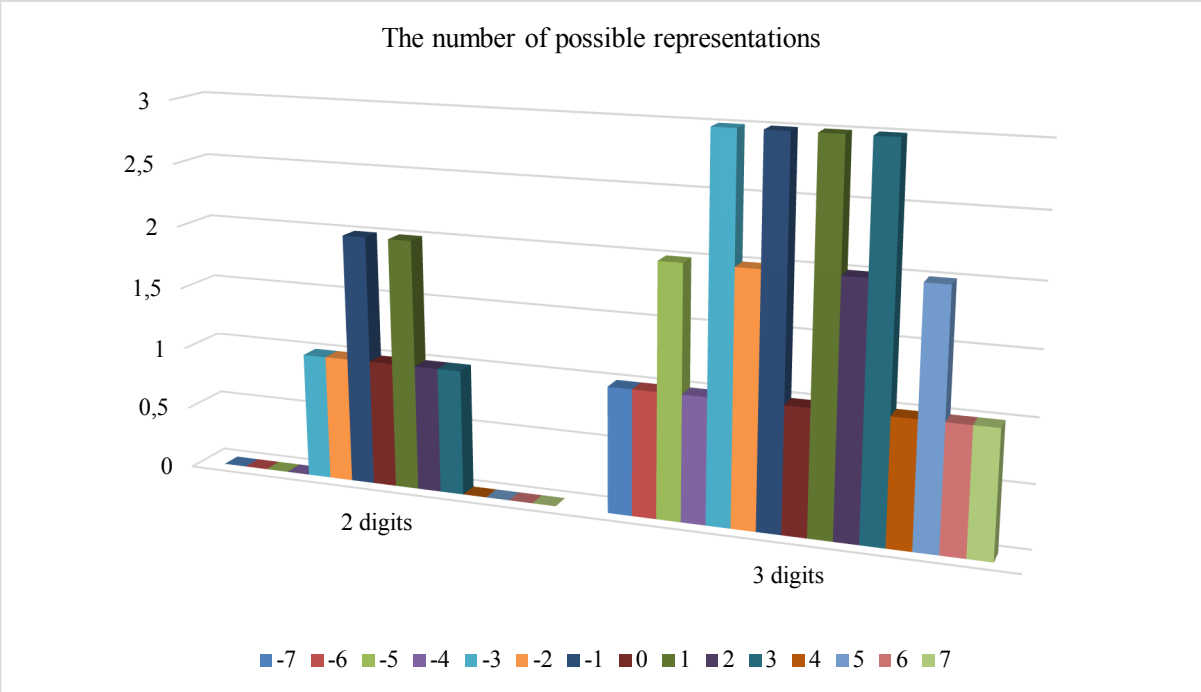

Fig. 5. Count of numbers representations on excess code with different digital capacity

Synthesis of topology using excess code. The synthesis process is similar, but it differs that digit T now is used in the process of insertion/replacement. In case of quasi-quantum topology, it used as an additional variant of insertion on shift. In case of hypercube: there are two variants of exchange for every digit, not only one. Table 5 shows de Bruin topology synthesis based on 2-digit excess code. Table 6 shows similar synthesis for hypercube.

Table 5

De Bruin topology synthesis for 2-digital excess code

Value	**Code**	**Put 0 to released bit**		**Put 1 to released bit**		**Put T to released bit**	
		Shift left	Shift right	Shift left	Shift right	Shift left	Shift right
0	00	00	00	01	10	0T	T0
1	01	10	00	11	10	1T	T0
-1	0T	T0	00	T1	10	TT	T0
2	10	00	01	01	11	0T	T1
3	11	10	01	11	11	1T	T1
1	1T	T0	01	T1	11	TT	T1
-2	T0	00	0T	01	1T	0T	TT
-1	T1	10	0T	11	1T	1T	TT
-3	TT	T0	0T	T1	1T	TT	TT

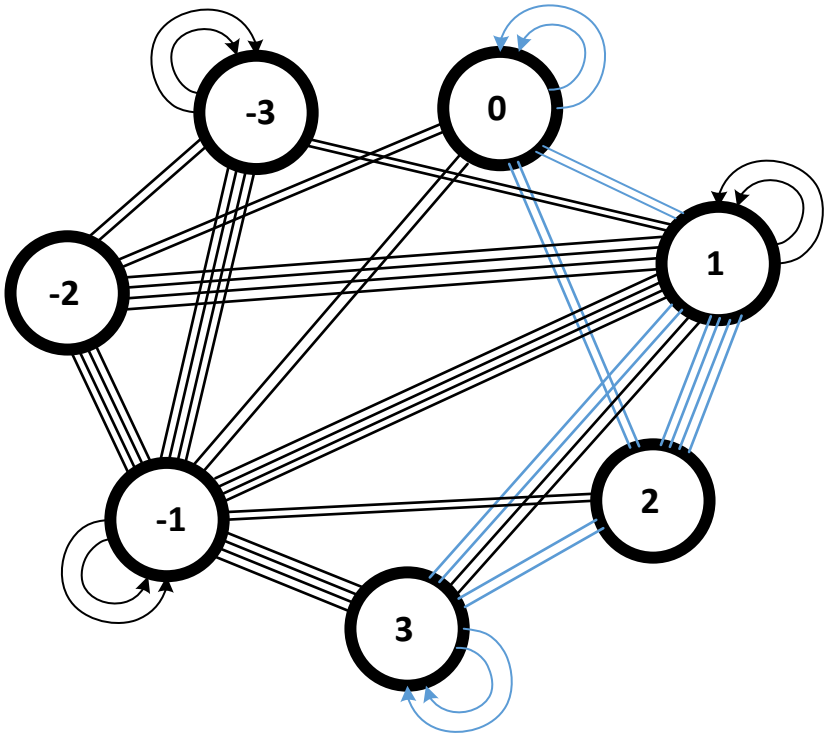

Fig. 6. De Bruin topology based on excess code, N=2

Table 6

Hypercube topology synthesis for 2-digital excess code

Value	**Code**	Exchange 1st bit		Exchange 2nd bit	
0	00	10	T0	01	0T
1	01	11	T1	0T	00
-1	0T	1T	TT	00	01
2	10	T0	00	11	1T
3	11	T1	01	1T	10
1	1T	TT	0T	10	11
-2	T0	00	10	T1	TT
-1	T1	01	11	TT	T0
-3	TT	0T	1T	T0	T1

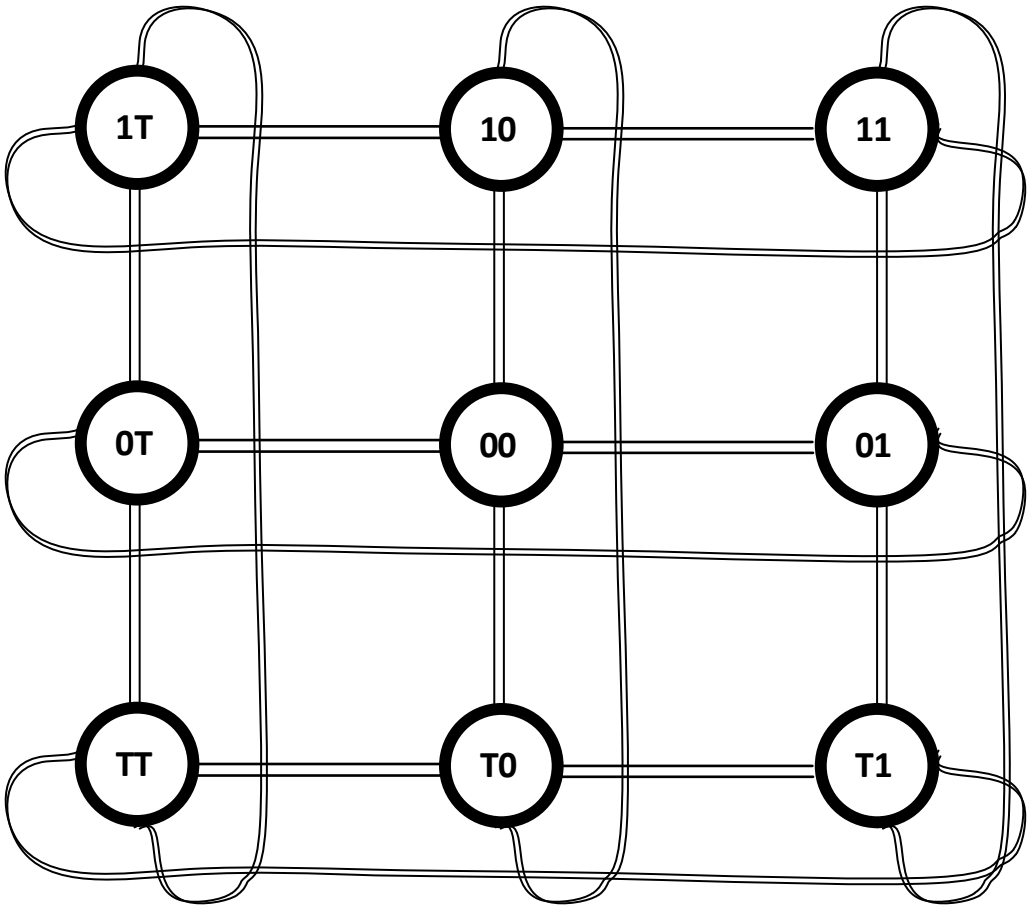

Fig. 7. Hypercube topology based on excess code, N=2

Using redundancy. There are several ways of using the proposed redundancy. Firstly, existing of several nodes with same numbers allows uniting those nodes in a cluster. This allows, to hide their essence from the user. In case of a node failure, another element in a cluster can take it's role. Also, the encoding allows access to each element separately. It allows to increase the system's performance in case of failure-free working through the use of additional elements.

The alternative way of using redundancy is a fault-tolerant routing [6-7], based on routing trees. The top of the tree is a node with same values of digits in the code. Those trees are constructed as follows: in case of quasi-quantum topology node's code is shifting left with insertion. In case of hypercube one digit in the code is inverted. If K-th digit been inverted on the previous step, on the next step we can invert only lower digits relative to K.

Usage of routing trees is follows: the routing program analyzes the next node on a route and if this node is busy or faulty, the transition on an alternative tree is performing. Thus, the task of bypassing the faulty nodes is solved simply and effectively. Also, this approach helps to decrease the probability of locking data transfer. If a needed node is busy while data transfer, this method allows to find free node using a transition on an alternative tree [8].

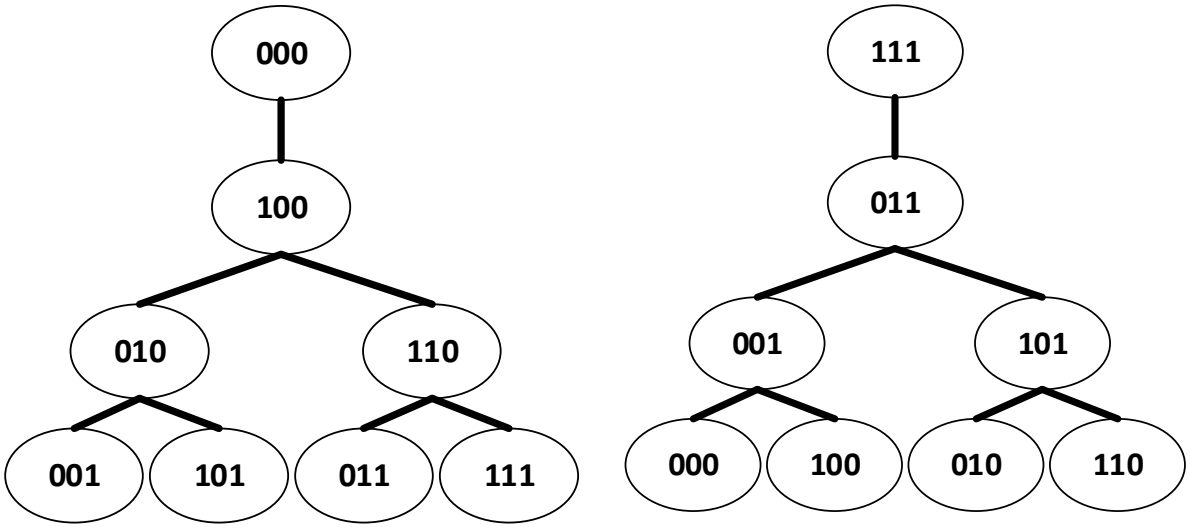

Fig. 8. Routing trees for classic de Bruin topology

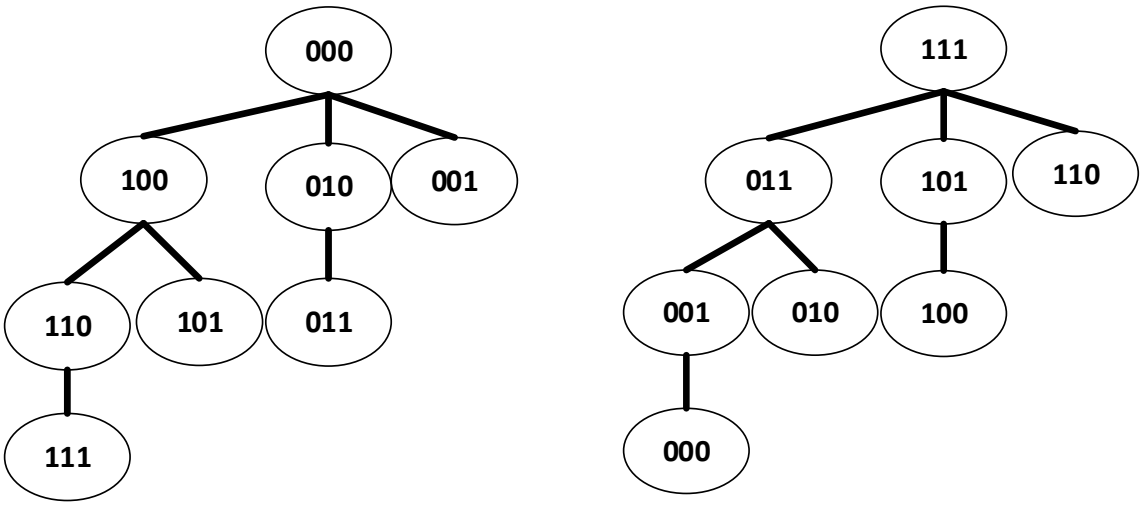

Fig. 9. Routing trees for classic hypercube topology

Of course, in case of failure of two and more nodes the malfunction bypassing can be difficult or even impossible to do, using the routing trees. In the case of fault-tolerant topologies, there are three elements with the same digits in code, not only two. As result, the 3 routing trees can be designed, and failure of 2 elements will not lead to system failure. Moreover, if elements with the same number have been substituted by clusters or multiprocessor systems, then it is possible usually to bypass the malfunction, by delivering information not through "original" node. In this situation, it is possible to use a node with the same number but with a different code.

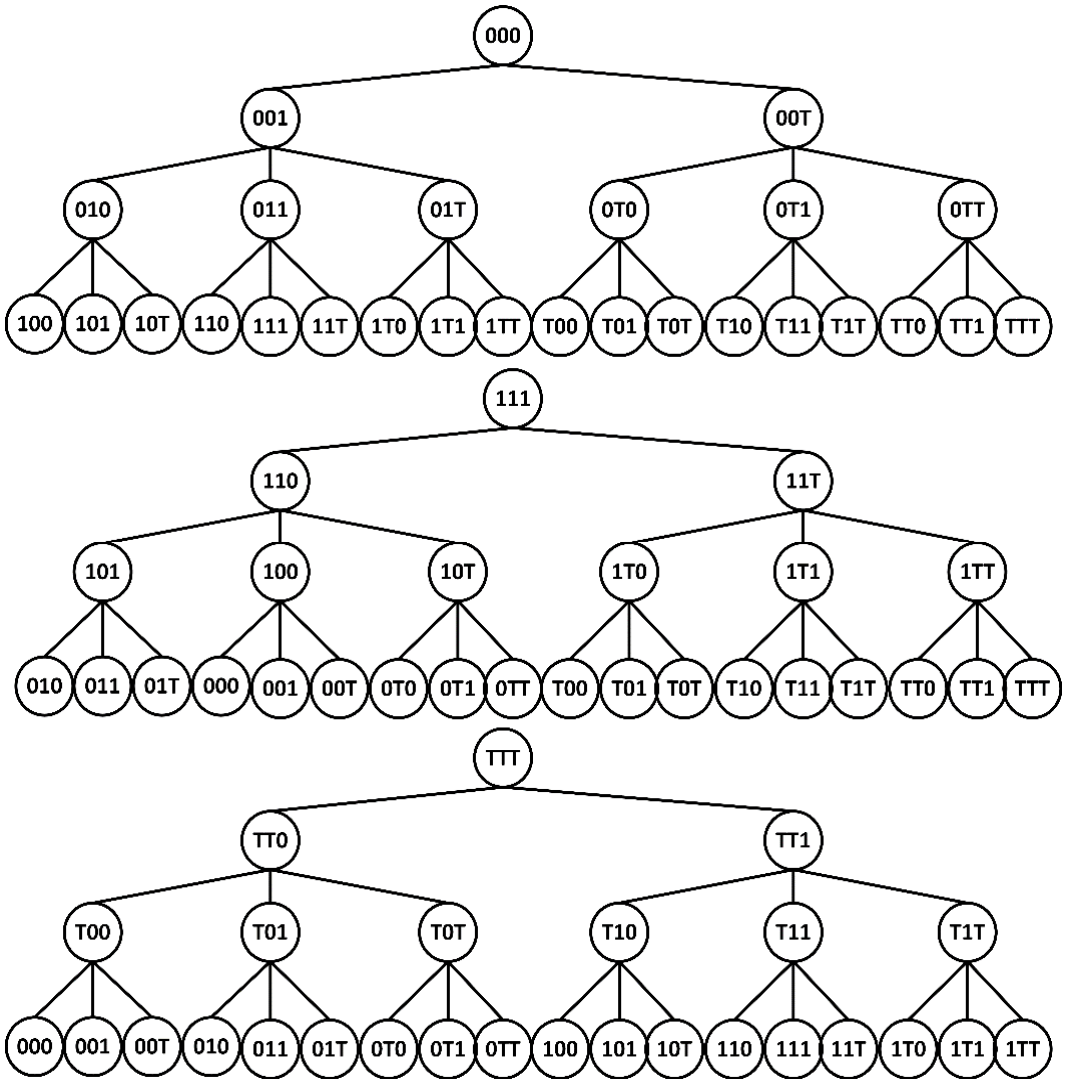

Fig. 10. Routing trees for de Bruin topology, based on 3-digits excess code

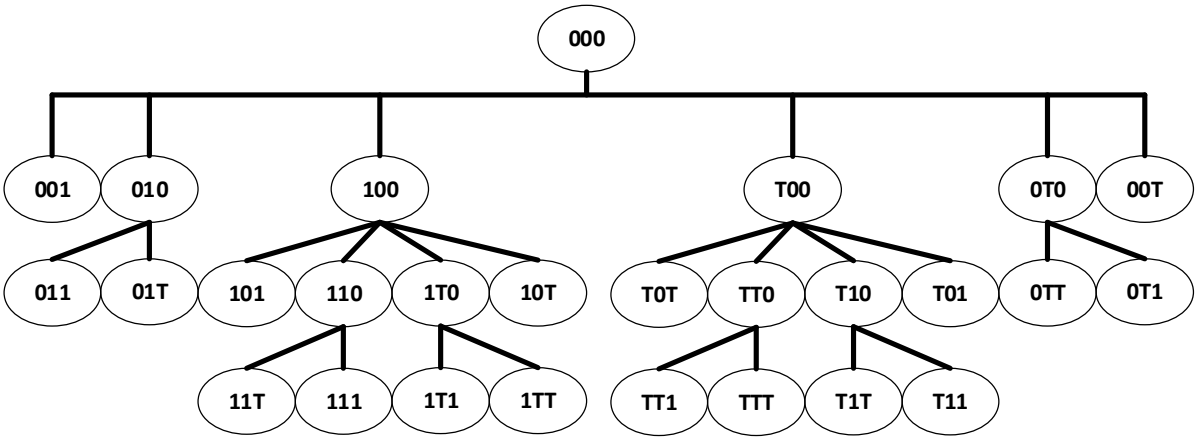

Fig. 11. Routing tree for hypercube, based on 3-digits excess code, built from node 000

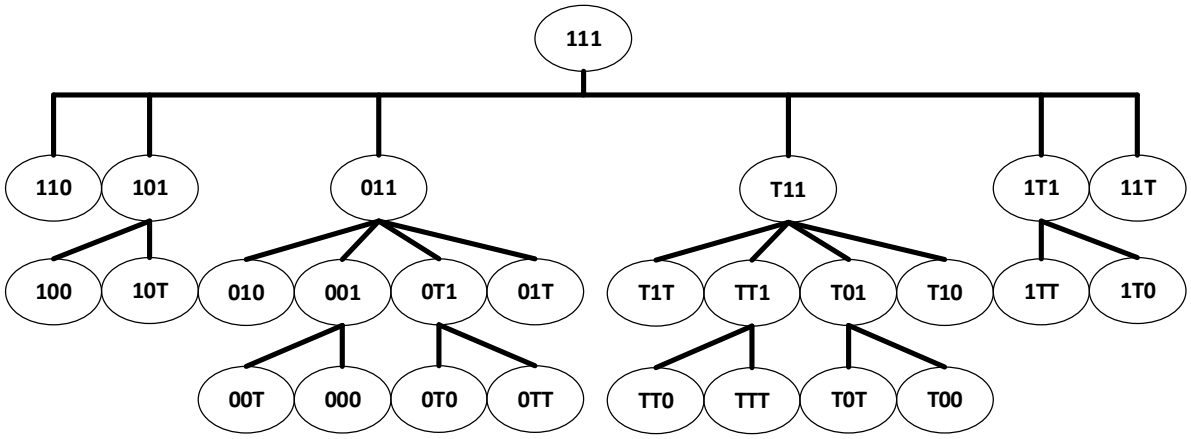

Fig. 12. Routing tree for hypercube, based on 3-digits excess code, built from node 111

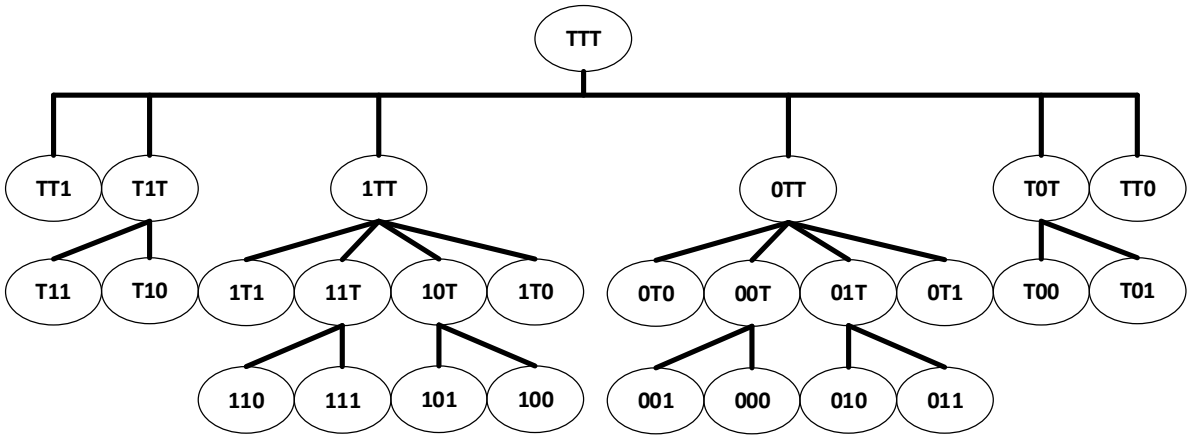

Fig. 13. Routing tree for hypercube, based on 3-digits excess code, built from node TTT

Comparison of designed topologies characteristics. The comparison of designed topologies was done. In table 7 describes basic parameters of topological structures such as power, diameter and others are given.

Table 7

Topologies comparison by the main characteristics

Parameter	**Base topologies**						**Redundant topologies**					
Topology	De Bruin			Hypercube			De Bruin			Hypercube		
Count of bits (N)	3	5	8	3	5	8	2	3	5	2	3	5
Count of nodes	8	32	256	8	32	256	9	27	243	9	27	243
Diameter	3	5	8	3	5	8	2	3	5	2	3	5
Power	4	4	4	3	5	8	6	6	6	4	6	10
Count of edges	16	64	512	12	80	1024	27	81	723	18	81	1215
Minimum number of nodes that must fail for the impossibility of using routing trees	2	2	2	2	2	2	3	3	3	3	3	3

Conclusions. In this paper method for fault-tolerant topology synthesis was proposed. It is based on using the excess code. Tree-based routing method considered, that allows to bypass the faulted nodes.

The advantages of proposed topologies are: a higher level of fault tolerance, which is achieved with the using of an additional tree, and the possibility of substitution nodes with same numbers in clusters, that additionally increases the fault tolerance of system and allows at the same time to abstract from the structure of clusters and access to concrete elements. It prevents the inaction of elements and associated loss of performance.

However, proposed solutions has disadvantages too. Firstly, it is the power increasing. As can be seen from the table, power of de Bruin topology is always equal to 4, while power for hypercube is equal to the number of bits in the code of nodes. Excess code usage increases the power of both topologies. For the de Bruin topology, power become equal to 6 and it isn't depending of the number of nodes. In case of usage excess encoding for hypercube, the power increases as 2N, where N is the number of digits of the encoding. It means that if N = 4 the power of hypercube will be equal to 8, while the optimal for realization is the value of this parameter from 4 to 6. As a result, the complexity of this topology increases with large N.

There are several possible options to improve the solutions developed. First, if necessary, it is possible to reduce the number of elements, by assigning to the same element different codes with the same value. Second, it is possible to remove some connections that are not used in trees, thereby reducing the cost on designing the system. Thirdly, it is possible to duplicate the key elements of trees, thereby increasing the fail-tolerance of the entire system.

References

1. InfiniBand Trade Association. *InfiniBand Architecture Specification*, 1, 1.2.1. Retrieved from http://www.infinibandta.com.
2. Eric Gamess, Humberto Ortiz-Zuazaga (2016). Low Level Performance Evaluation of InfiniBand with Benchmarking Tools. *International Journal of Computer Network and Information Security (IJCNIS)*, 8(10), 12-22.
3. Mukhin, V., Volokyta, A., Heriatovych, Y., Rehida, P. (2018). Method for efficiency increasing of distributed classification of the images based on the proactive parallel computing approach. *Advances in Electrical and Computer Engineering*, 18(2), 117-122.
4. Hu, Z., Mukhin, V., Kornaga, Y., Volokyta, A., & Herasymenko, O. (2017). The scheduler for distributed computer systems based on the network centric approach to resources control. In: Proceedings of the 2017 *IEEE 9th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, IDAACS 2017 (pp. 518-523).
5. Linder D., Harden J. (1991). An adaptive and fault tolerant wormhole routing strategy for k-ary n-cubes. *IEEE Transactions on computers,* 1, 2-12.
6. Lionel M. Ni, Philip K. (1993). A Survey of Wormhole Routing Techniques in Direct Networks. *Computer,* 2, 62-76.
7. Richard J. Cole, Bruce M. Maggs, Ramesh K. Sitaraman. (2001). On the Benefit of Supporting Virtual Channels in Wormhole Routers. *Journal of Computer and System Sciences*, 62(1), 152-177.
8. Washington N., Perros H. (2007). Performance Analysis of Traffic-Groomed Optical Networks Employing Alternate Routing Techniques. *Lecture Notes in Computer Science*, 4516, 1048-1059.

УДК 004.052.3:004.22

Георгій Луцький, Артем Волокита, Павло Регіда, Олександр Гончаренко

ВИКОРИСТАННЯ НАДЛИШКОВОГО КОДУ ДЛЯ ПОБУДОВИ ВІДМОВОСТІЙКИХ ТОПОЛОГІЙ

Актуальність теми дослідження. *Завдання підвищення відмовостійкості є одним із головних завдань при побудові обчислювальної системи і при модернізації вже існуючої. Особливу увагу їй приділяють при побудові мультикомп'ютерних систем чи кластерів. Найбільш цікавими способами збільшення відмовостійкості є використання топологічної структури системи для обходу несправності та використання того чи іншого елемента системи для заміщення несправного. Звісно, це потребує розробки специфічної топології. У статті розглянуто розробку відмовостійких версій популярних топологій, таких як квазіквантова та гіперкуб, на основі надлишкового коду 0/1/-1.*

Постановка проблеми. *Важливою частиною будь-якої мультикомп'ютерної системи є її топологічна структура. Від неї залежить маршрутизація повідомлень у системі, швидкість передачі повідомлень і відмовостійкість. У статті запропоновано метод для підвищення відмовостійкості, що ґрунтується на використанні надлишкового коду.*

Аналіз останніх досліджень і публікацій. *Нині добре описано синтез таких топологій, як гіперкуб чи топологія де Бруйна, є роботи, що розглядають методи збільшення відмовостійкості за допомогою дублювання обчислювальних елементів. Також наявні роботи, що розглядають використання маршрутизації на основі дерев для покращення відмовостійкості системи.*

Виділення недосліджених частин загальної проблеми. *Не розглянутими на сьогодні залишаються можливості застосування надлишкового кодування 0/1/-1 для створення нових відмовостійких топологій на основі вже існуючих.*

Постановка завдання. *Завданням є опис синтезу відмовостійких топологій, розгляд можливостей використання їхніх особливостей та аналіз основних характеристик у порівнянні між собою та з класичними версіями на основі двійкового коду.*

Виклад основного матеріалу. *Описано синтез гіперкуба та топології де Бруйна на основі звичайного двійкового коду та надлишкового коду 0/1/-1, розглянуто можливості використання надлишковості, передусім для підвищення відмовостійкості, проведено порівняльний аналіз усіх згаданих топологій.*

Висновки відповідно до статті. *Виконано аналіз характеристик, виділено основні переваги та недоліки запропонованих топологічних структур, висунуто пропозиції щодо їх покращення.*

Ключові слова: *відмовостійкість; де Бруйн; гіперкуб; надлишковий код.*

Рис.: 13. Табл.: 7. Бібл.: 9.

Loutskii Heorhii – Professor, Department of Computer Engineering**,** National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute" (37 Pobedy Av., 03056 Kyiv, Ukraine).
Луцький Георгій Михайлович – професор, кафедра обчислювальної техніки, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського» (просп. Перемоги, 37, м. Київ, 03056, Україна).
E-mail: georgijluckij80@gmail.com

Volokyta Artem – associate professor, Department of Computer Engineering**,** National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute" (37 Pobedy Av., 03056 Kyiv, Ukraine).
Волокита Артем Миколайович – доцент, кафедра обчислювальної техніки, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського» (просп. Перемоги, 37, м. Київ, 03056, Україна).
E-mail: artem.volokita@kpi.ua

Rehida Pavlo – assistant, Department of Computer Engineering**,** National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute" (37 Pobedy Av., 03056 Kyiv, Ukraine).
Регіда Павло Геннадійович – асистент, кафедра обчислювальної техніки, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського» (просп. Перемоги, 37, м. Київ, 03056, Україна).
E-mail: pavel.regida@gmail.com

Goncharenko Olexandr – student, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute" (37 Pobedy Av., 03056 Kyiv, Ukraine).
Гончаренко Олександр Олексійович – студент, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського» (просп. Перемоги, 37, м. Київ, 03056, Україна).
E-mail: alexandr.ik97@ukr.net

Loutskii, H., Volokyta, A., Rehida, P., Goncharenko, O. (2019). Using excess code to design fault-tolerant topologies. *Technical sciences and technologies*, *1* (15), 134–144.