A Method of Developing a Checkable Self-Recovery Floating-Point Pipeline System

Oleksandr Drozd[1][0000-0003-2191-6758], Igor Kovalev [2][0000-0001-6065-2893], Oleksandr Martynyuk [3][0000-0002-3043-5924], Kostiantyn Zashcholkin [4][0000-0003-0427-9005], Julia Drozd[5][0000-0001-5880-7526]

Odessa National Polytechnic University, Ave. Shevchenko 1, 65044, Odessa, Ukraine
drozd@ukr.net, igoryan33@ua.fm, anmartynyuk@ukr.net, const-z@te.net.ua, yuliia.drozd@opu.ua

Abstract. The objective nature observed in the development of the computer world makes it possible to predict requests for the development of promising computer systems. The parallelism and fuzziness of the natural world encourages the development of pipeline systems for the processing of approximate data. The progressive development of high-risk objects increases the requirements for safety-related systems in improving the checkable and fault-tolerant solutions used in them. A method for developing a checkable fault-tolerant floating-point pipeline system is proposed. Matrix circuits of pipeline sections are unified in the form of lines of identical elements. The operands and floating-point results obtain an additional bit excluded from the calculations. Its movement creates many versions of calculation execution, ensuring the checkability of circuits and their fault tolerance in the form of self-recovery when selecting a version that excludes a faulty element. The method uses the synergy of different types of performed functions to simplify the system and to counter faults.

Keywords: safety-related system, approximate data, floating-point pipeline system, version, checkability, fault tolerance, self-recovery.

1 Introduction and Related Works

The development of computer systems demonstrates a number of objectively evolving patterns that are appropriate to take into account to advance promising solutions. We offer as such a promising solution the development of a Checkable Self-Recovery Floating-Point Pipeline System, which can provide high-performance processing of approximate data in the field of critical applications. This article aims to show the urgent need for such a multifaceted system and the method of its development.

Related works can be divided into two main groups.

The first group orients the proposed system to perform approximate calculations with their matrix and pipeline parallelization. These works explain the increasing role of approximate calculations [1-3] and demonstrate their priority execution in floating-

point formats [4, 5], where the mantissa bits are diversified by their division into most and least significant ones [6-8], and the errors caused in these bits by faults become essential and inessential to the trustworthiness of the calculated results. Logic in parallelization of calculations is shown in software products [9, 10] and development of green technologies [11, 12]. Low efficiency of matrix structures [13, 14] is analyzed in the use of operating elements forming them [15, 16] and power consumption [17-19].

The second group of works forms the basis for adaptation of the developed system to its use in safety-related applications [20, 21]. Fault-tolerant solutions [22, 23], including correction codes and reconfiguration [24, 25], do not guarantee functional safety in case of lack of checkability, which is better known as testability [26, 27]. For safety-related systems, lack of checkability leads to the hidden fault problem [29, 30], better known for the accident consequences of unsuccessful attempts to search such faults [31, 32]. The identification of this problem as a growth challenge [33, 34] related to the established practice of using matrix structures [35, 36] defines solutions, including the use of multi-version technologies [37, 38].

In addition, the proposed method uses the features of truncated operations [39]. Program implementation of the method is performed for iterative array multiplier [40, 41].

A more detailed analysis of the references used is consistent with the purpose of the paper in justifying the need to develop the proposed system and is discussed in the following sections.

A rest of the paper has a following structure. Section 2 examines the state of the art and explains why the computer system is most in demand to handle approximate data in floating-point formats and why it is constructed as a pipeline using matrix circuits in its sections. In addition, Section 2 justifies the feasibility to counteracting failures in the Self-Recovery System and the need to develop a Checkable System for the use in safety-related applications. Section 3 outlines the basic provisions of a method developing the Checkable Self-Recovery Floating-Point Pipeline System and analyzes its properties to justify such a name. Section 4 shows the case study of the proposed method on the example of an iterative array multiplier of mantissas as a section of a pipeline system.

2 Main Features of High-Demand Computer System

Patterns observed in the development of computer systems can be viewed in terms of a resource-based approach that analyzes the integration of the computer world into the natural one [1].

The computer world created by human largely repeats the development of the natural world, but in a shorter time frame. Observation of analogy suggests a single basis in the development of these worlds and dominance of objective processes, the direction of which determines the vector of development. Integration into the natural world takes place by structuring under its features. The computer world exhibits to the highest degree two such features: parallelism and fuzziness. The mass production of personal computers, which has been going on for decades, allows us to trace the process of structuring to these features. Hardware support for approximate computing began

with an optional Intel 287/387 coprocessor. In the Pentium processor, we see the emergence of FPU (Floating-Point Unit) pipelines to handle approximate data with increased performance. The modern graphics processor contains thousands of such pipelines, which are used for parallel computing using CUDA (Compute Unified Device Architecture) technology [2, 3]. Following the development vector is encouraged by outstanding achievements in key indicators: performance and memory, which have simultaneously increased millions of times from KHz to GHz and from MB to TB.

It is important to note that no one planned such development of personal computers. The natural integration process was indicative of the primacy of objective processes. A particular conclusion is that computer systems should be developed in the direction of processing approximate data and, above all, in floating-point formats that have become most common [4, 5].

The resource-based approach addresses the performance challenges needed to complete all work in a limited amount of time, generate reliable results, and invest resources in performance and trustworthiness. These resources are models, methods, and means. The resource-based approach shows three levels of resource development, from replication to diversification and self-sufficiency. Replication is observed in the absence of conflicts with the natural world, i.e. in open resource niches where integration takes place by stamping clones. This process is based on productivity, which aims to exceed fertility over mortality. Filling resource niches leads to a crisis of overproduction in which only those who show features survive. They are transformed from clones to individuals, versions, and thus are raised to the level of diversification, where integration is achieved through trustworthiness, i.e. adequacy to the natural world [1, 6].

Note that floating-point formats diversify the mantissa bits by dividing them into most and least significant bits [7, 8]. Faults cause errors in these bits that are essential and inessential to the trustworthiness of the result.

Today's computer world is experiencing a replication boom. The software is composed of installed modules. They can be huge in size and contain only one operator needed, but will be connected, thanks to the open resource niches of memory and performance of computer systems [9, 10]. Mobile, autonomous systems are characterized by a limited niche in energy supply and therefore rise to the level of diversification using green technologies [11, 12].

Hardware solutions are based on matrix structures that also reflect the level of replication. Arithmetic blocks contain parallel shifters and adders, iterative array multipliers and dividers performing operations in parallel codes [13, 14]. Matrix structures are inefficient and resource-intensive. For example, an iterative array multiplier performing a key operation of approximate calculations in one clock cycle comprises G^2 operational elements, $2G - 2$ of which are connected in series, where G is the size of the operand. In the cases of $G = 32$ and $G = 64$, each operating element is used by 1.6% and 0.8%, respectively [15, 16]. The rest of the clock cycle time is filled with parasitic transitions of signals, which occur due to different length of their propagation paths [17, 18]. The number of these glitches is many times more than the number of functional transitions. The dynamic and static components of energy consumption are mainly determined by parasitic transitions and large dimensions of matrix structures.

Pipelining improves the performance of calculations by paralleling them at the level of diversification of operations executed in pipeline sections. However, sections of the modern pipeline system contain single-cycle matrix circuits with all the problems caused by the replication level [18, 19].

It should be noted that matrix structures exhibit the greatest problems in the field of critical applications, where computer systems are transformed into safety-related systems. These systems are aimed at ensuring functional safety of high-risk facilities to prevent accidents [20, 21]. Quantitative and qualitative growth of high-risk facilities, including powerful power plants and power networks, various types of weapons, determines the improvement of safety-related systems as a priority in the development of information and computer technologies for successful integration into the natural world.

According to current international standards, functional safety of safety-related systems is based on the use of fault-tolerant solutions [22, 23]. These solutions provide for different types of redundancy and reconfiguration designed to parry a certain number of failures, usually one or two [24, 25].

However, this fault tolerance is insufficient when the checkability is insufficient. Its most simple form, known as testability, shows dependence only on the structure of the digital circuit, i.e. it is a structural checkability [26, 27]. On-line testing detects errors if the input data shows faults of the digital circuit, that is, within the framework of structural-functional checkability, depending also on the input data [28, 29].

Resource niches are particularly prone to closure in the field of safety-related applications. Computer systems become safety-related by diversifying the operating mode, which is divided into normal and emergency modes. In matrix structures, such separation is inherited by input data, which become different in these modes and diversify structural-functional checkability, creating the problem of hidden faults. They can be accumulated in normal mode, which does not use the input data that displays them, and cause many failures when the input data changes in emergency mode. Their number may exceed capabilities of any fault-tolerant system [30].

The problem of hidden faults is solved in practice by means of imitation modes, which recreate emergency conditions and repeatedly led to them due to human factor or unauthorized activation by fault [31, 32]. The resource-based approach identifies the problem of hidden faults as a challenge of growth, when the system in checkability rises to the level of diversification, and its components continue to be stamped based on matrix structures relating to the level of replication [33, 34].

This interpretation of the problem determines the ways to solve it by developing components to the system level. Decision bluntly is based on reduction of matrix structures. However, matrix structures have prevailed for several decades and have created strong infrastructure in their support, including models, methods and modern CAD supported by libraries of ready-made solutions [35, 36].

However, we can see many other opportunities in raising resources to the level of diversification, including ways that maintain the established tradition of using matrix circuits in pipeline system sections. We can propose the use of multi-version technologies that have already been widely used in safety-related systems to counter common cause failures [37].

Typically, the number of versions in multi-version systems is limited due to their increasing complexity. Systems with two versions have become most common, which must be as independent as possible to resist common cause failures. It should be noted that version redundancy is common in the natural world, but it is characterized by a high degree of version connectivity, which increases the functionality of the system with its limited resources. For example, we can observe the high functionality of the fingers of the hand.

Intersecting versions are not an obstacle to counteracting failures in three or more versions, as is the case in computer systems with strongly connected versions [38]. As will be shown later, these systems combine fault tolerance with high checkability and low hardware costs that decrease as the number of versions increases. Fault tolerance is ensured by self-recovery by eliminating the fault from the computing process.

Thus, following the development vector in the development of a promising system leads us to the next choice. Parallelization of calculations in the processing of approximate data at the level of diversification is expedient to implement into the pipeline floating-point system. The field of safety-related applications dictates the need to combine fault tolerance and checkability of circuit solutions that can be implemented on the basis of a checkable self-recovery system with strongly connected versions.

3 Main Provisions of the Suggested Method

We consider a pipeline system whose sections perform floating-point arithmetic operations in parallel codes based on matrix structures. The developed pipeline system is aimed at completing the entire task in case of faults of not more than one in each section.

Typically, matrix structures combine arrays of uniform operational elements with regular connections. The parallel shifter consists of a line of multiplexers, and the parallel adder contains a line of full adders. The iterative array multiplier and divider comprise two-dimensional matrices of identical operational elements and their structures can also be considered as lines of rows or columns of the operational elements. This similarity, seen in arithmetic circuit structures, allows them to be unified as a line of generally series-connected elements. This structure, consisting of *M* elements, is being supplemented by another element and is transformed in a ring which serves as the basis for constructing the pipeline section of the strongly connected version system shown in Fig. 1.

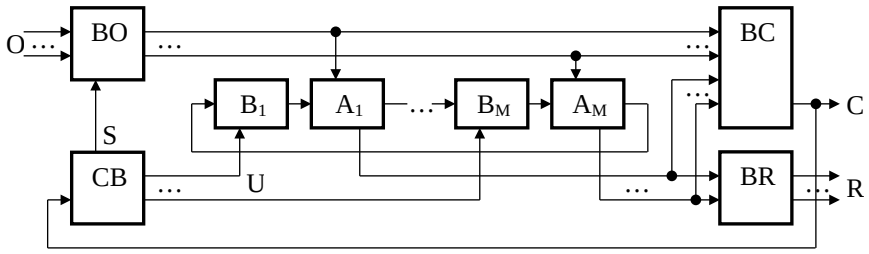

Fig. 1. Strongly connected version system

The system performs an arithmetic operation in one clock cycle of the pipeline section and comprises an operating block OB in the form of a ring, as well as control block CB, checking block BC, blocks BO and BR of operands and result.

The ring consists of $M + 1$ elements A: A_1, …, A_M and $M + 1$ units B: B_1, …, B_M, which can break the ring between any adjacent elements. A ring break converts it to a line of elements with one extra element located at the last position. Thus, the system can be in one of the $M + 1$ states, in each of which contains the version of the original line and its addition by an extra element. The failure of this element is blocked because the code it calculates is not involved in the operation.

The CB block comprises a modulo M + 1 counter storing the state code S, which is the version number. The BC block checks the result and changes the S code to the next value in case of error. This code is supplied to control inputs of BO and BR blocks, and after conversion to unitary code U – to control inputs of B_1, …, B_M units. Operands and result contain M bits and zero-bit at extra additional position. The blocks BO and BR comprise (M + 1)-bit shifters controlled by the S code. The shifters select operands and result according to the current version. Operands are selected from the inputs O of the multiplier and supplied to the inputs of the OB and BC blocks. The result is selected from the outputs of the OB block and supplied to the inputs of the BC block and the output R of the multiplier.

The logic of the system operation consists in changing its state in each clock cycle, i.e. for each execution of arithmetic operation. If an error is detected, the state continues to change to a value where the fault is at the position of the additional element not involved in the calculations, and the BC block reports no error. This state is fixed until the entire job performed by the pipeline system is completed.

Note that the presence of the additional position excludes from the calculations not only the extra element A, but also the extra elements in the $(M + 1)$ bit shifters, i.e. in the state change schemes. The fault of the B unit is also masked, both in case of failure to break the ring and otherwise.

The permanent state change makes the system checkable even with minimal operand changes.

The developed system performs various functions in accordance with its classification as pipeline, checkable, self-recovery and processing floating-point data. We can develop synergies between these types of functions towards simplifying the system and countering failures.

Floating-point operations are performed with mantissas and exponents, interaction between which is carried out by renormalization of operands and normalization of results using arithmetic shifters of mantissas. The combined implementation of these functions with the cycle shift functions performed in the discussed self-recovery system ensures their simplification in FPGA project by 23% and 46% for 16- and 32-bit mantissas, respectively [38].

The permanent change of states of the checkable system in combination with its pipeline structure allows simplifying the cyclic shift function to move no more than one position. This shift is performed with respect to the previous state due to its storage by the pipeline registers. In addition, this minimum amount of shift is combined

with the movement of mantissa when the multiplication and division results are normalized.

As is known, mantissa processing can be performed by truncated multiplication and division with elimination of the lower part in the matrix of operational elements from calculations without reduction of single accuracy [39]. For 16- and 32-bit mantissas, the lower part is 30% and 37% of the total matrix, respectively, and it is an area of fault masking when performing complete multiplication. The use in the BC block of the inequality checking method will allow to ignore errors caused by faults in the lower part of the matrix [6, 18]. Thus, the self-recovery system receives not only an extra element, but also a significant portion of the matrix to exclude a fault from the calculations in multiplication and division of mantissas.

System recovery can occur over several clock cycles with some loss of trustworthiness of results or performance in the event of a stop of the input data stream. It is advantageous to shift the operands towards decreasing the amount of error.

4 Case Study of the Proposed Method

The experimental verification of the proposed method was executed using on an example of iterative array Braun multiplier [40] using its program model developed on the freely distributed Delphi 10 Seattle demo version [41].

The program sets a random sequence of operands and simulates the execution of multiplication in case of short circuit between two points of the scheme of a randomly selected operating element in the structure of an n-bit iterative array multiplier for $n = 8, ..., 15$. The structure of the multiplier is supplemented by an additional string and column and therefore contains a matrix of $(n + 1)\times(n + 1)$ operational elements consisting of an AND element and a full adder. Circuit points (inputs of AND element, as well as inputs and outputs of adder) are selected randomly to inject fault.

The main program panel is shown in Fig. 2.

The panel shows binary and decimal values for operand codes A, B and the Cm and C results which are calculated and simulated, respectively.

The operational elements of additional rows and columns are highlighted in yellow, as are the corresponding additional bits of operands and results. The fault masking area is painted blue and the older half of the results are painted green.

The occurrence of a fault in the form of an error initiates a change in the state code by one and a cyclic shift of both operands by one position towards the higher bits. In this case, the conjunction, which is calculated and processed by the faulty operation element, reduces the weight of the caused error up to the transition from the highest bit to the lowest one.

Results were checked by inequalities taking into account that for the normalized mantissas m_A and m_B, $1 \le m_A < 2$, $1 \le m_B < 2$, the product m_C is limited to inequality of $(m_A + m_B) - 4 < m_C < 2 \min(m_A, m_B)$ [18].

During the simulation, the program counts the number of operations performed and the number of results calculated with an error, an essential error, and an essential error detected, and determines the probabilities of these errors. In addition, the program

calculates the number of Te and Tc clock cycles that it took to correct the error and error detected by the inequality checking method.

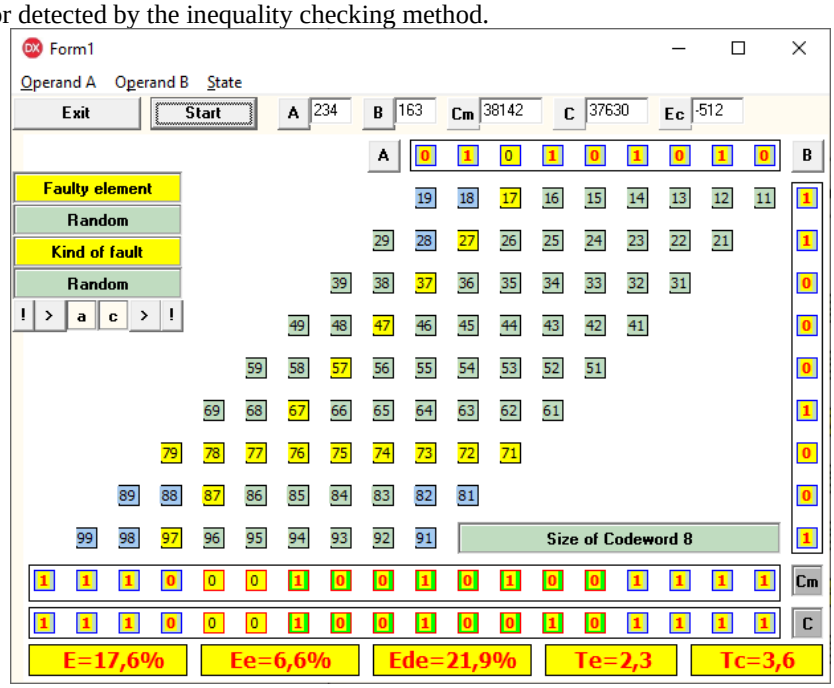

Fig. 2. Main panel of the program

Experiments conducted in order to increase the size of *n* from 8 to 15 showed an increase in the error probability, the probability of an essential error, and a decrease in the probability of a detected essential error from 17.6% to 22.8%, from 6.6% to 11.1% and from 21.9% to 9.9%, respectively.

The dependence of the averaged number of clock cycles Te and Tc on the operand size is shown in Fig. 3.

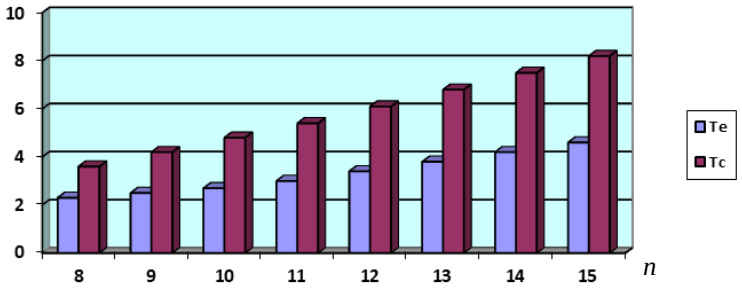

Fig. 3. Dependence of clock cycles Te and Tc on the operand size

The average number of clock cycles Te and Tc increases from 2.3 to 4.6 and from 3.6 to 8.2, i.e. in the rate of increase in the size of operands. It should be noted that the undetected essential error is halved in the next clock cycle and therefore will not be

detected by the inequality checking. If the fault continues to produce an error, it will first become inessential and then essential and detectable when it moves to the highest bits of the product. This clock cycle will start the schema recovery process.

5 Conclusions

The development of the computer world demonstrates a dominant objective component, the following of which is an important prerequisite for the design of promising computer systems. Structuring resources under the parallelism and fuzziness of the natural world advances the development of pipeline systems for processing approximate data. Filling and closing resource niches encourage the development of high-risk facilities and safety-related systems with increased functional safety requirements, which is based on the use of checkable fault-tolerant solutions. Thus, there is a need to develop checkable fault tolerant floating-point pipeline systems.

The proposed method of developing such systems is based on the dominance of matrix structures in modern computer design. The use of matrix circuits in pipeline sections allows them to be unified by presenting identical elements in the form of lines. This structure of sections creates conditions for their conversion into checkable fault-tolerant solutions based on strongly connected versions. The structure of the operands and the results of the floating-point operations are given an additional bit excluded from the calculations. Moving an additional bit creates multiple versions of performing calculations on the same line of elements, ensuring the checkability of circuits and their fault tolerance when selecting a version that excludes a faulty element from the computing process.

As preferences due following to objective processes, the method gains synergy of different types of functions justifying the name of the system, and uses this harmony to simplify the solution and counter faults. Inherent in floating-point operations, data renormalization and normalization, as well as the search for the true version, is based on the use of shifters and therefore allows jointly implementing these functions and simplifying the system. Similarly, and for the same goal, the implementation of functions ensuring the system pipelining and checkability is combined. Synergy of features in checking functions and floating-point data processing, manifested in truncated operations, provides additional masking of faults in emergency mode of safety-related systems.

References

1. Drozd, J., Drozd, A., Antoshchuk, S.: Green IT engineering in the view of resource-based approach. In: Green IT Engineering: Concepts, Models, Complex Systems Architectures, SSDC, vol. 74, Springer, Berlin, pp. 43-65 (2017) doi: 10.1007/978-3-319-44162-7_3
2. NVIDIA CUDA Compute Unified Device Architecture. Programming Guide / Version 1.0, NVIDIA Corporation (2007)
3. Andrecut, M.: Parallel GPU implementation of iterative PCA algorithms. Journal of Computational Biology, vol. 16, no. 11, pp. 1593-1599 (2009). Online. [Available]: http://dx.doi.org/10.1089/cmb.2008.0221

4. IEEE Std 754™-2008 (Revision of IEEE Std 754-1985) IEEE Standard for Floating-Point Arithmetic. IEEE 3 Park Avenue New York, NY 10016-5997, USA (2008)
5. Synopsys DWFC Flexible Floating-Point Overview. no. August, pp. 1-6 (2016)
6. Drozd, J., Drozd, A., Al-dhabi, M.: A resource approach to on-line testing of computing circuits. In: IEEE East-West Design & Test Symposium, Batumi, Georgia, pp. 276-281 (2015) doi: 10.1109/EWDTS.2015.7493122
7. Kekre, H.B., Mishra, D., Khanna, R., Khanna, S., Hussaini A.: Comparison between the basic LSB Replacement Technique & Increased Capacity of Information Hiding in LSB"s Method for Images. International Journal of Computer Applications. 45, No. 1, 33-38 (2012)
8. Zashcholkin, K., Ivanova, O.: The control technology of integrity and legitimacy of LUT-oriented information object usage by self-recovering digital watermark. In: CEUR Workshop Proceedings, vol. 1356, pp. 498-506 (2015)
9. Pomorova, O., Savenko, O., Lysenko, S., Kryshchuk, A., Bobrovnikova, K.: A technique for the botnet detection based on DNS-traffic analysis. In: Gaj, P., Kwiecień, A., Stera, P. (eds.) CN 2015. CCIS, vol. 522, pp. 127-138. Springer, Heidelberg (2015)
10. Hovorushchenko, T, Pomorova, O. Ontological approach to the assessment of information sufficiency for software quality determination. SEUR-WS, vol. 1614, pp. 332-348 (2016)
11. San Murugesan, Gangadharan, G.R.: Harnessing Green IT. Principles and Practices, UK: Wiley and Sons Ltd. (2012)
12. Vakaliuk, T., Antoniuk, D., Morozov, A. et. al.: Green IT as a tool for design cloud-oriented sustainable learning environment of a higher education institution. E3S Web of Conferences 166, 10013 (2020)
13. Palagin, A., Opanasenko, V.: The implementation of extended arithmetic's on FPGA-based structures. In: IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems, vol. 2, Bucharest, Romania, pp. 1014-1019 (2017)
14. Chernov, S., Titov, S., Chernova, L. et. al.: Algorithm for the simplification of solution to discrete optimization problems. Eastern-European Journal of Enterprise Technologies 3 (4), 1-12 (2018)
15. Ehliar, A.: Area efficient floating-point adder and multiplier with IEEE754 compatible semantics. In: International Conference on Field-Programmable Technology, pp. 131-138 (2014)
16. Drozd, O., Antoniuk, V., Nikul, V., Drozd, M.: Hidden faults in FPGA-built digital components of safety-related systems. In: 14th IEEE International Conference TCSET, Lviv-Slavsko, Ukraine, pp. 805-809 (2018) doi: 10.1109/TCSET.2018.8336320
17. Kumar Vasantha, Murthy Sharma N. S., Lal Kishore K.: A Technique to Reduce Glitch Power during Physical Design Stage for Low Power and Less IR Drop. In International Journal of Computer Applications (0975 – 8887), vol. 39, 18, 62-67 (2012)
18. Drozd, A., Antoshchuk, S., Drozd, J. et. al.: Checkable FPGA Design: Energy Consumption, Throughput and Trustworthiness. In: Green IT Engineering: Social, Business and Industrial Applications, SSDC, vol. 171, Springer, Berlin, pp. 73-94 (2019) doi: 10.1007/978-3-030-00253-4_4
19. Li, H.F.: A structural study of parallel pipelined systems. In: PhD Dissertation, Univ. of California., Berkeley (1975).
20. Smith D., Simpson K.: The Safety Critical Systems Handbook, 4th Edition, Butterworth-Heinemann (2016)
21. Ivanchenko, O., Kharchenko, V., Moroz, B. et. al.: Risk Assessment of Critical Energy Infrastructure Considering Physical and Cyber Assets: Methodology and Models. In: 10th IEEE International Conference IDAACS, Lviv, Ukraine, pp. 225-228 (2018)
22. International Electrotechnical Commission, Nuclear Power Plants: Instrumentation and Control for Systems Important to Safety – General Requirements, Rep. IEC 61513, IEC, Geneva (2001)

23. Romankevich, A., Feseniuk, A., Romankevich, V., Sapsai, T. About a fault-tolerant multiprocessor control system in a pre-dangerous state. In: 9th IEEE International Conference DESSERT, Kyiv, Ukraine, pp. 215-219 (2018) doi: 10.1109/DESSERT.2018.8409129
24. Yatskiv, V., Yatskiv, N., Jun, S. et. al.: The use of modified correction code based on residue number system in WSN. In: 7th IEEE International Conference IDAACS, Berlin, Germany, pp. 513-516 (2013) doi: 10.1109/IDAACS.2013.6662738
25. Donthi, S, Haggard, R.L.: A survey of dynamically reconfigurable FPGA devices. In: 35th Southeastern Symposium on System Theory, Morgantown, WV, USA, pp. 422-426 (2003).
26. Matrosova, A., Nikolaeva, E., Kudin, D., Singh, V.: PDF testability of the circuits derived by special covering ROBDDs with gates. In: IEEE East-West Design and Test Symposium, Rostov-on-Don, Russia, pp. 1-5 (2013)
27. Kondratenko, Y.P., Kozlov, O.V., Topalov, A.M., Gerasin, O.S. Computerized system for remote level control with discrete self-testing. In: CEUR Workshop Proceedings Open Access, vol-1844, pp. 608-619 (2017) http://ceur-ws.org/Vol-1844/10000608.pdf
28. Coppad, D., Sokolov, D., Bystrov, A., Yakovlev, A.: Online Testing by Protocol Decomposition. In: 12th IEEE International On-Line Testing Symposium, Como, Italy, pp. 263-268 (2006) doi: 10.1109/IOLTS.2006.45
29. Drozd, A., Drozd, J., Antoshchuk, S. et. al.: Objects and Methods of On-Line Testing: Main Requirements and Perspectives of Development. In: IEEE East-West Design & Test Symposium, Yerevan, Armenia, pp. 72-76 (2016) doi: 10.1109/EWDTS.2016.7807750
30. Drozd, O., Kuznietsov, M., Martynyuk, O., Drozd, M.: A method of the hidden faults elimination in FPGA projects for the critical applications. In: 9th IEEE International Conference DESSERT, Kyiv, Ukraine, pp. 231-234 (2018) doi: 10.1109/DESSERT.2018.8409131
31. Gillis, D.: The Apocalypses that Might Have Been. [Online]. Available: http://www .popmech.ru/go.php?url=http%3A%2F%2F www.damninteresting.com%2F%3Fp%3D913.
32. U.S.-Canada Power System Outage Task Force: Final Report on the August, 14, 2003 Blackout in the United States and Canada: Causes and Recommendations USA (2004)
33. Drozd, O., Kharchenko, V., Rucinski, A. et. al.: Development of Models in Resilient Computing. In: 10th IEEE International Conference on Dependable Systems, Services and Technologies, Leeds, UK, pp. 2-7 (2019) doi: 10.1109/DESSERT.2019.8770035
34. Kulanov, V., Kharchenko, V., Perepelitsyn, A.: Parameterized IP Infrastructures for fault-tolerant FPGA-based systems: Development, assessment, case-study. In: IEEE East-West Design and Test Symposium, St. Petersburg, Russia, pp. 322-325 (2010)
35. Intel Quartus Prime Standard Edition User Guide: Getting Started. [Online]. Available: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug-qpsgetting-started.pdf, last accessed 2019/03/20.
36. Xilinx LogiCORE IP floating-point operator v7.0, product guide, PG060 (2014). [Online]. Available: www.xilinx.com/support/documentation,
37. Kharchenko, V.: Multy-version Systems: Models, Reliability, Design Technologies. In: 10th European Conference on Safety and Reliability, pp. 73-77, Munich, Germany (1999)
38. Drozd, O., Kovalev, I., Drozd, M. et. al.: Sharing of Functional and Special Means in Pipeline Floating-Point Systems with Strongly Connected Versions. In: 11th IEEE International Conference IDAACS, Metz, France, pp. 249-253 (2019) doi: 10.1109/IDAACS.2019.8924289
39. Park, H.: Truncated Multiplications and Divisions for the Negative Two's Complement Number System. In: Ph.D. Dissertation. The University of Texas at Austin, Austin, USA (2007).
40. Neeraja, B., Sai Prasad Goud, R.: Design of an area efficient Braun multiplier using high speed parallel prefix adder in cadence. In: IEEE International Conference on Electrical, Computer and Communication Technologies, Coimbatore, India (2019)
41. Delphi 10 Seattle: Embarcadero (2015) https://www.embarcadero. com/ru/products/delphi