A Method of Hidden Faults Opposition for FPGA-Based Components of Safety-Related Systems

Oleksandr Drozd[1][0000-0003-2191-6758], Vitaliy Romankevich[2][0000-0003-4696-5935], Alexei Romankevich[3][0000-0001-5634-8469], Mykola Kuznietsov[4][0000-0002-3043-5924], Myroslav Drozd[5][0000-0003-0770-6295]

[1,4,5]Odessa National Polytechnic University, Ave. Shevchenko 1, 65044, Odessa, Ukraine
[2,3]National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Victory avenue, 37, Kyiv, Ukraine
[1]drozd@ukr.net, [2,3]romankev@scs.kpi.ua, [4]koliaodessa@ukr.net, [5]myroslav.drozd@opu.ua

Abstract. The paper is devoted to the problem of hidden faults, which is inherent in safety-related systems aimed at ensuring the functional safety of high-risk facilities to counter accidents. The problem of hidden faults is considered from the perspective of a resource-based approach as a problem of growth from a lower level of replication to the next level of diversification in the development of models, methods and means. Computer systems in critical applications have risen to the level of diversification in the division of the operating mode into normal and emergency, in the input data and structurally functional checkability, which for digital components have become different in these modes. Digital components continue to be traditionally stamped based on matrix structures that reflect the level of replication. The lag of the components from the development of the system creates a problem of hidden faults which can be accumulated during the normal mode and reduce the fault tolerance of the components and the functional safety of the system in emergency mode. We propose a method of counteracting hidden faults by raising components to the level of diversification in the promising field of FPGA designing. The proposed method uses the natural version redundancy inherent in the program code of the FPGA projects with LUT-oriented architecture. The method generates and selects versions of the program code, reducing many hidden faults of short circuits between neighboring inputs of LUT units. Possible hidden faults are eliminated by increasing the checkability of the FPGA project in normal mode and the trustworthiness of the results calculated in emergency mode.

Keywords: safety-related system, digital component, FPGA project, LUT unit, program code, hidden fault, resource-based approach, growth problem, checkability, trustworthiness, version redundancy

1 Introduction and Related Works

The most important direction in the development of information technologies is their improvement in the field of critical applications. This area, which is exceptional in nature, has already acquired a scale commensurate with the living space of mankind. Power grids and power plants, high-speed ground and air transport, hazardous chemical production, space technology and weapons are high-risk facilities [1, 2].

Risk assessment includes two factors related to the probability of an accident and the cost of its consequences. The factor of emergency consequences has a tendency of constant growth, which is caused by development of objects of high risk in the direction of their complication and increase of capacity, as well as increase of their number, density of location and proximity to densely populated areas [3, 4].

Only reducing the probability of accidents can counteract the growth of this factor. This mission is devoted to information technology implemented in safety-related systems, which, according to international standards, are aimed at "ensuring functional safety of both the system and the control object for preventing accidents and reducing the consequences if they occur" [5, 6].

The challenge in achieving functional safety "is to design the system in such a way as to prevent dangerous failures or to control them when they arise" [7]. This provision automatically includes hidden faults in the concept of "dangerous failures", as they pose a problem of their control for safety-related systems and their components [8, 9].

The fault control function is related to the development of the concept of checkability. At an early stage, this concept is known as testability or structural checkability of the digital circuit, characterizing it from the point of view of simplicity in the development of tests for fault detection [10, 11].

The structural nature of testability is due to the fact that it is completely determined by the structure of the digital circuit. The next stage in the development of checkability is related to on-line testing of digital circuits [12, 13], where checkability also shows dependence on input data, which characterizes it as structurally functional. The definition of self-testing circuits formulated in the theory of designing the totally self-checking circuits is known [14, 15]. Self-testing circuits are evaluated in checkability taking into account input data [16, 17].

Safety-related systems make a significant contribution to the development of the concept of checkability of the digital components by diversifying the operating mode into normal and emergency. Inputs are also included in the diversification process and become different in these modes. Then the structurally functional checkability of the digital circuit is converted into a double-mode, which differs in normal and emergency operation due to different input data. Such development of checkability converts harmless hidden faults of conventional computers working only in one operating mode, into dangerous ones. Indeed, a fault hidden throughout the operating mode will have no effect on the functionality of a regular computer. A hidden fault becomes a problem for safety-related systems because such faults can be accumulated in normal mode without causing errors under conditions of insufficient structurally functional checkability of the circuit, i.e. lack of necessary input data. In the emergency mode, the checkability is enriched with new input data and creates conditions for the manifestation of accumulated faults in the form of errors [18, 19].

The problem of hidden faults has a long history known from unsuccessful attempts to detect these faults using imitation modes aimed at recreating emergency conditions. The activation of such modes often provides for the shutdown of emergency protections, which has become one of the causes of the Chernobyl disaster.

In addition, history knows many examples of emergency consequences as a result of unauthorized activation of imitation modes because of human factor or due to the resulting fault [20, 21].

The presence of imitation modes creating a real danger to functional safety can be explained by two reasons:

- the high significance attached to hidden faults, which are feared more than emergency conditions created by imitation mode;
- lack of confidence in the fault tolerance of the solutions used, on which functional safety of critical systems and control objects is built [22-24].

The resource-based approach, which explores the integration of the artificial world created by human into the natural one, refers to the problem of hidden faults as a growth challenge. This approach identifies three levels in the development of models, methods and means: replication, diversification and self-sufficiency as a development goal. It shows the transition of safety-related systems to diversification and the backlog of their components, which continue to be stamped at a lower level of replication using matrix structures. Such classification of the problem of hidden faults determines the ways of its solution by raising the components to the system level [25, 26].

One of the most important directions in the development of digital components for critical systems is associated with FPGA designing. FPGA technologies are also a prime example of replication level dominance. FPGA chips contain Configurable Logic Blocks or Logical Elements, prepared iterative array multipliers, and chains for rapid carry propagation to add parallel codes, the libraries of IP-cores with matrix structures [27, 28].

However, FPGA refers to programmable hardware that raises stamped element matrices to the level of diversification by using the natural version redundancy in program code of FPGA projects with LUT-oriented architecture [29-31].

In this architecture, the computing process is organized using LUT units that are logic function generators. Their arguments arrive at the inputs of the LUT unit. The description of the logical function is stored in the memory of the LUT unit and written to this memory in the process of programming the FPGA project as program code. In the case of four inputs A, B, C and D, the memory of the LUT unit contains $2^4 = 16$ bits [32, 33].

Versions of the program code are created for each pair of the LUT units where output of the first LUT unit is connected to the input of the second one. The signal between the LUT units may be transmitted by a direct or inverse value using one of two versions of the program code. The inverse value at the output of the first LUT unit of the pair is provided by inverting the bits of its memory and changing its program code accordingly. The obtained inversion at the input of the second LUT unit of the pair is compensated by changing the program code with replacement bits of the memory [34].

This form of redundancy has been used to increase the trustworthiness of the calculated results by generating program code versions and selecting the best one from the position of masking the faults between neighboring LUT unit inputs of the FPGA project [35, 36].

The selection of the version with the best structurally function checkability of the LUT units of the FPGA project in the normal mode of the safety-related system or trustworthiness of the results calculated in the emergency mode is proposed in [37].

A disadvantage of both solutions using version redundancy of program code is simulation of calculations on all normal and emergency mode inputs for each combination of versions generated by the LUT units of the FPGA project. The number of program code versions is defined as 2^Z, where Z is the number of first LUT units of all pairs.

For example, in the case of $Z = 30$ and $Z = 60$, simulation of calculations performed in the FPGA project is repeated for each input word of each mode more than 10^9 and 10^{18} times, respectively. The number of inputs is defined as 2^U, where U is the number of inputs of the simulated scheme of the FPGA project. For $U = 20$, the simulation must be repeated 10^{15} and 10^{24} times. Such a large number of iterations significantly limits the capabilities of the method in the processing of complex circuits. In addition, the choice of versions that increase the checkability of the FPGA project in normal mode or the trustworthiness of the results with the onset of emergency mode helps to reduce a set of the hidden faults, but in general is not the best solution.

We offer a method to counter hidden faults of the FPGA project taking into account the peculiarities of this kind of faults. The method allows maximum use of version redundancy of program code to reduce many possible hidden faults. Section 2 contains the main provisions of the proposed method. Section 3 describes the case study of the method using the FPGA project on example of the iterative array multiplier.

2 Main Provisions of the Suggested Method

The proposed method uses the synergy of several types of natural version redundancy. First of all, the method takes into account the natural version redundancy of safety-related systems, which is evident in their designing for operation in two essentially different modes: normal and emergency.

In addition, the method uses version redundancy of hidden fault elimination solutions. They pose a danger to fault tolerant decisions while meeting two conditions: accumulation of faults during normal mode and their manifestation in the form of errors in emergency mode. Therefore, the hidden fault is eliminated if at least one of the above conditions is excluded. Thus, the resistance to hidden faults can be achieved with the use of two versions of the solution, which consists in improving the checkability of LUT units and the trustworthiness of the results calculated on them, respectively.

The fault of the short circuit between the two neighboring inputs of the LUT unit also demonstrates natural version redundancy. It consists of two fault states: its masking or error manifestation in the case of the same signal values at the neighboring inputs of the LUT unit and otherwise, respectively. Fault masking increases the trustworthiness of the calculated results, and its manifestation in the form of error improves the checkability of the LUT unit.

Versions of the program code allow to control the input of the second LUT unit. They are assigned to this input directed or inverse value. Change version manages the state of the fault, showing it or masking for improving checkability of the LUT unit or the trustworthiness of the results. These improvements can be achieved concurrently, assigning them to different modes: normal and emergency, respectively.

The method generates and considers all versions of the program code that can be created for the second LUT unit of each pair. For example, if only two of the four LUT unit inputs connected to the outputs of the previous LUT units, then these LUT units will form two pairs and 4 versions.

The method distinguishes between three sets M_N, M_E and $M_{N\&E}$ of bits in the memory of each LUT unit: bits addressable in the normal, emergency, and in both modes, respectively. Faults which cause errors in bits of both M_N and $M_{N\&E}$ sets are not hidden, as can be detected in the normal mode. Therefore, the fault of the short circuit may be hidden, but if they cause errors only in bits of the M_E set and do not contain errors in bits, addressed in the normal mode.

The fault circuit between neighboring inputs of the LUT unit maintains proper access to its memory bits for identical values of these inputs and indicates the remaining bits in the values corresponding to zero values of the signals.

Examples of the effect of the short circuit faults on accessing the memory of the LUT unit are shown in Fig. 1.

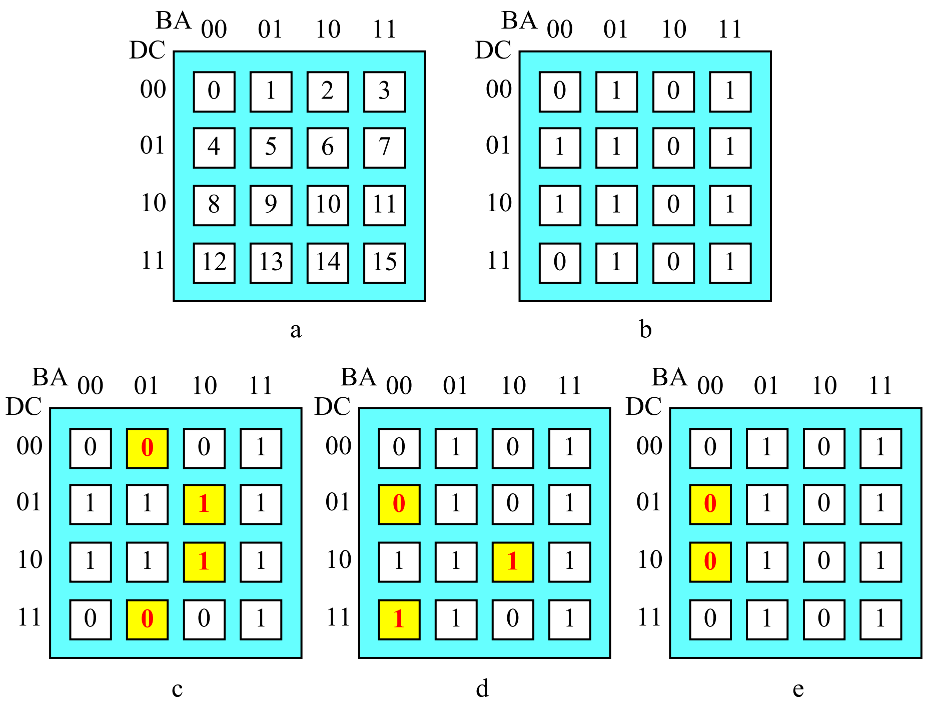


Fig. 1. Examples of LUT unit memory: numbers (a) and values (b) of bits and

memory in cases of shorts between A and B (c), B and C (d), C and D (e) inputs

The numbering of the LUT unit bits and the correct values of bits of memory, which is programmed with the ABBA$_{16}$ code, shown in Fig. 1, a and b. The memory of the LUT unit for cases of shorting inputs A and B, B and C, C and D is shown in Fig. 1, c, d, e, respectively.

Shorting the A and B inputs copies the values of bits BA 00 column into the memory array columns BA 01 and BA 10. BA 11 column bits retain their value. Shorting the B and C inputs copies the value of bits located at the intersection of column BA 00 and BA 01 with lines DC 00 and DC 10 into bits at the intersection of column BA 00 and BA 10 with a DC 01 and DC 11 lines as well as into bits at the intersection of column BA 10 and BA 11 with lines DC 00 and DC 10. The bits located at the intersection of columns BA 10 and BA 11 with lines DC 01 and DC 11 retain their values. Shorting the C and D inputs copies the values of bits of the DC 00 line into lines DC 01 and DC 10. Bits of the DC 11 lines retain their values. Erroneous bit values are highlighted in yellow.

The method performs the following steps:

Step 1: Determination of all second LUT units of the circuit and for each of them the set of all versions of program code.

Step 2. Simulation of calculations executed in FPGA project for all of the input data, i.e., U times, with determination of the $M_{NE} = M_N \cup M_{N\&E}$ and M_E bit sets in memory for each second LUT unit of the pair.

Step 3. Determination of the M_{NE} and M_E sets of bits in the memory for each version of each second LUT unit of the pair.

Step 4. Determination of all possible faults of the short circuit between neighboring inputs of each second LUT unit of the pair for cases where at least one of these inputs is connected to the output of the previous LUT unit.

Step 5. Determination of the program codes in the view, distorted under the influence of any faults. These program codes are generated for each version of each second LUT unit of pairs and compared with the correct program code versions. The erroneous bits detected in the sets M_{NE} and M_E, form the sets $M_{NE\ ER}$ and $M_{E\ ER}$, respectively.

Step 6. The FPGA project program code is generated using versions containing the minimum number of bits in $M_{E\ ER}$ sets with $M_{NE\ ER} = \varnothing$.

Step 7: The resulting program code is compared with the initial and least successful in the number of bits in the $M_{E\ ER}$ sets with $M_{NE\ ER} = \varnothing$ to evaluate the capabilities of the method.

The method improves the checkability of the FPGA project in the normal mode, and thus eliminates the hidden faults by choosing versions with sets of $M_{NE\ ER} \neq \varnothing$. In case $M_{NE\ ER} = \varnothing$, the method selects the versions with the lowest number of bits in the $M_{E\ ER}$ set to reduce the set of hidden faults that manifest themselves in an emergency mode.

Reducing many hidden faults, performed in both modes, aimed at improving the trustworthiness of the results calculated in the emergency mode. From this position, the trustworthiness of the FPGA project results with respect to hidden faults of the short circuit between neighboring inputs of LUT units can be estimated taking into account erroneous memory bits addressed only in emergency mode in the case of $M_{NE\ ER} = \varnothing$.

The trustworthiness of the result read from the output of the LUT unit can be estimated as $T_{LUT} = (1 - (A_{E\ ER} / (3A_{E\ ER}))\ K_{NE\ ER}) \times 100\%$, where $A_{E\ ER}$ and A_E are the number of bits in the $M_{E\ ER}$ sets for all three types of short circuit and in the M_E set, respectively, $K_{NE\ ER} = 1$ if $M_{NE\ ER} = \varnothing$, and $K_{NE\ ER} = 0$ otherwise.

The trustworthiness of the FPGA project results can be estimated by the arithmetic average of the T_{LUT} values calculated for all LUT units.

The contribution that is made to the trustworthiness by the checkability of the LUT units in the normal mode can be estimated similarly taking into account the error memory bits of the $M_{E\ ER}$ set in the case of $M_{NE\ ER} \neq \varnothing$.

Comparison of the best solution with the initial project and the least successful version of the program code shows the effectiveness of the method in the specific example of FPGA design and the potential of the method, respectively.

3 Case Study of the Proposed Method

Experimental verification of the method was carried out using CAD Quartus Prime 18.1 Lite Edition on the example of a study of a 4-bit iterative array multiplier implemented in Intel Max 10 FPGA 10M50DAF672I7G [38, 39]. The digital circuit of the obtained FPGA project contains 8 inputs which are supplied with 4-bit operands, 30 LUT units with four inputs for performing the multiplication operation, and 8 product outputs. The digital circuit simulation was carried out using the program implementation of the method. The program was developed in the free Delphi 10 Seattle demo version [40].

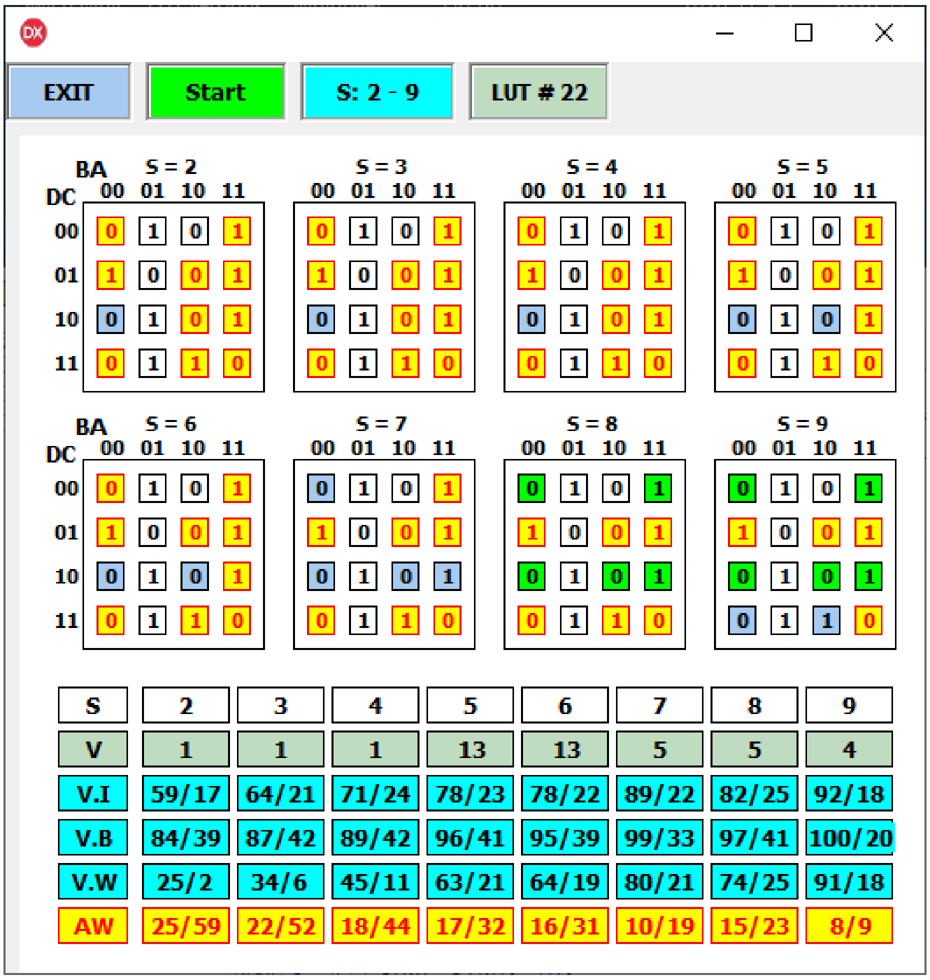
As initial data, the program uses a description of a digital circuit with an indication of the connections of its inputs or outputs of previous LUT units to the inputs of each next LUT unit and the outputs of the circuit. In addition, the number of digital circuit inputs, the number of LUT units and their program codes are indicated.

The program presents the results of examining the digital circuit on the main panel, which is shown in Fig. 2.

The main panel is invoked by pressing the «Start» key and allows to complete the program on «Exit» command. The main panel allows to view the values of the memory bits for all LUT units operating at different threshold values S, dividing the input data of the normal and emergency modes.

The values of the factors smaller than the S threshold refer to the normal mode, and the rest to emergency one. The "S: 2 - 9" key determines 8 threshold values from 2 to 9. Each press of this key shifts the range of S values by one to the maximum: "S: 8 - 15" and then the value "S: 2 - 9". The "LUT # 22" key specifies the number 22 of the considered LUT unit. Clicking on this key allows to proceed to the LUT unit with the following number. The largest number is replaced by the first. Bits addressable in the normal and in emergency mode only, colored in green and yellow, respectively. Blue color indicates addressing in both modes.

The lower part of the panel shows the results of the proposed method for each value of the S threshold. Line "V" shows the decimal number of the best version of the program code. The binary code dcba$_2$ of the version number determines the inverted inputs of the second LUT unit by the unit values of the corresponding bits.



S = 2

DC \ BA	00	01	10	11
00	0	1	0	1
01	1	0	0	1
10	0	1	0	1
11	0	1	1	0

S = 3

	00	01	10	11
	0	1	0	1
	1	0	0	1
	0	1	0	1
	0	1	1	0

S = 4

	00	01	10	11
	0	1	0	1
	1	0	0	1
	0	1	0	1
	0	1	1	0

S = 5

	00	01	10	11
	0	1	0	1
	1	0	0	1
	0	1	0	1
	0	1	1	0

S = 6

DC \ BA	00	01	10	11
00	0	1	0	1
01	1	0	0	1
10	0	1	0	1
11	0	1	1	0

S = 7

	00	01	10	11
	0	1	0	1
	1	0	0	1
	0	1	0	1
	0	1	1	0

S = 8

	00	01	10	11
	0	1	0	1
	1	0	0	1
	0	1	0	1
	0	1	1	0

S = 9

	00	01	10	11
	0	1	0	1
	1	0	0	1
	0	1	0	1
	0	1	1	0

S	2	3	4	5	6	7	8	9
V	1	1	1	13	13	5	5	4
V.I	59/17	64/21	71/24	78/23	78/22	89/22	82/25	92/18
V.B	84/39	87/42	89/42	96/41	95/39	99/33	97/41	100/20
V.W	25/2	34/6	45/11	63/21	64/19	80/21	74/25	91/18
AW	25/59	22/52	18/44	17/32	16/31	10/19	15/23	8/9

Fig. 2. The main panel of the program implementation of the suggested method

For example, version 13 = 1101$_2$ means inverting inputs D, C and A. Number 0 = 0000$_2$ indicates the preservation of the source program code [41].

The next three lines «V.I», «V.B» and «V.W» evaluate the trustworthiness of the calculated results to the initial, the best and the least successful version of the program code for FPGA project, respectively. The numerator includes the trustworthiness of the results calculated in the FPGA project, and the denominator contains the contribution that is made to the trustworthiness of the checkability of the LUT units in normal mode. The last line in the numerator and denominator shows the gain in the trustworthiness of the best solution compared to the initial project and the least successful version of the program code, respectively.

Diagrams of the dependence in trustworthiness of the results on the S threshold for the best, initial and least successful solution are shown in Fig. 3.

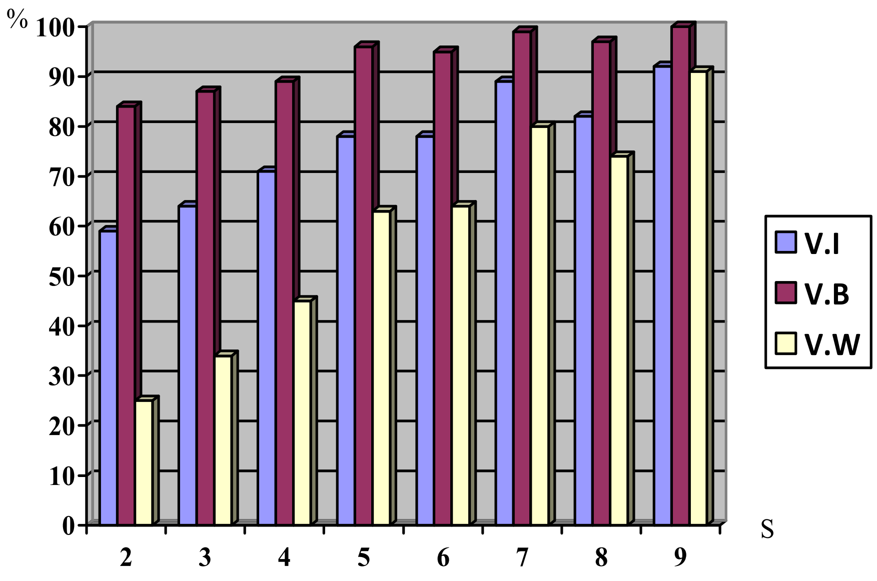


Fig. 3. Diagrams of the dependence in trustworthiness of the results on the S threshold for the best, initial and least successful solution

Diagrams show a tendency to increase trustworthiness of the results with an increase in the S threshold.

Therefore, it is important that the greatest gain in the best solution obtained by the proposed method is achieved for small threshold values that are typical for circuits operating in normal mode at a noise level. For S = 2 and S = 3, the trustworthiness of the results increases relative to the initial program code from 59% to 84% and from 64% to 87%, i.e. by 25% and 23%, respectively.

4 Conclusions

FPGA designing, which is a promising direction in the development of digital components for safety-related systems, opens up new possibilities for solving the hidden fault problem inherent in such systems. The programmability of FPGA projects allows to solve this problem as a growth challenge by raising the components to the level of diversification, where critical systems are located in operating modes, input data and digital circuit checkability.

The proposed method uses the version redundancy in the program code of FPGA projects with a LUT-oriented architecture to reduce the set of hidden faults in the short circuit of neighboring inputs of the LUT units.

Such faults distort the addressing of the memory bits of the LUT units. Memory bits read at the wrong address may have erroneous values that reduce the trustworthiness of the calculated results.

A hidden fault is dangerous with errors that are not detected in normal mode and distort the results in emergency one.

The method reduces the number of hidden faults in two ways. The first way is to search for each LUT unit the program code versions, showing a failure in the normal mode. The second way applies to LUT units for which there are no such version, and chooses the version with the least amount of erroneous values in bits, addressed only in emergency mode.

The advantage of the proposed method is its low complexity, limited not by the set of all versions of the FPGA project program code, but by the set of LUT units with mutually independent examination of their program codes.

Further research is planned in the direction of expanding the circle of problems which can be identified and solved as a growth challenge in relation to safety-related systems and their components based on the development and practical application of the concepts of checkability and natural version redundancy, in particular in FPGA designing.

References

1. International Atomic Energy Agency, Evaluation of the Status of National Nuclear Infrastructure Development, IAEA Nuclear Energy Series No. NG-T-3.2, IAEA, Vienna (2008)
2. Hiromoto, R. E., Sachenko, A., Kochan, V. et. al.: Mobile Ad Hoc Wireless Network for Pre- and Post-Emergency Situations in Nuclear Power Plant. In: WS 2014 - 2nd IEEE International Symposium on Wireless Systems within the Conferences on Intelligent Data Acquisition and Advanced Computing Systems, Offenburg, Germany, pp. 92-96 (2014) doi: 10.1109/IDAACS-SWS.2014.6954630
3. Ivanchenko, O., Kharchenko, V., Moroz, B. et. al.: Risk Assessment of Critical Energy Infrastructure Considering Physical and Cyber Assets: Methodology and Models. In: IDAACS 2018 - 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Lviv, Ukraine, pp. 225-228 (2018) doi: 10.1109/IDAACS-SWS.2018.8525594
4. Smith, D. J.: Reliability, Maintainability and Risk. Practical Methods for Engineers, 9th Edition, Butterworth-Heinemann (2017)
5. International Atomic Energy Agency, On-line Monitoring for Improving Performance of Nuclear Power Plants, Part 2: Process and Component Condition Monitoring and Diagnostics, IAEA Nuclear Energy Series No. NP-T-1.2, IAEA, Vienna (2008)
6. Smith D., Simpson K.: The Safety Critical Systems Handbook, 4th Edition, Butterworth-Heinemann (2016)
7. International Electrotechnical Commission, Nuclear Power Plants: Instrumentation and Control for Systems Important to Safety – General Requirements for Systems, Rep. IEC 61513, IEC, Geneva (2001)

8. Efanov, D., Lykov, A., Osadchy, G.: Testing of relay-contact circuits of railway signalling and interlocking. In: EWDTS 2017 - IEEE East-West Design and Test Symposium, Novi Sad, Serbia, pp. 242-248 (2017) doi: 10.1109/EWDTS.2017.8110095
9. Drozd, O., Antoniuk, V., Nikul, V., Drozd, M.: Hidden faults in FPGA-built digital components of safety-related systems. In: TCSET 2018 - 14th International Conference "Modern problems of radio engineering, telecommunications and computer science, Lviv-Slavsko, Ukraine, pp. 805-809 (2018) doi: 10.1109/TCSET.2018.8336320
10. Hahanov, V., Litvinova, E., Obrizan, V., Gharibi, W.: Embedded method of SoC diagnosis. Elektronika in Elektrotechn, no. 8, 3-8 (2008)
11. Matrosova, A., Nikolaeva, E., Kudin, D., Singh, V.: PDF testability of the circuits derived by special covering ROBDDs with gates. In: EWDTS 2013 - IEEE East-West Design and Test Symposium, Rostov-on-Don, Russia, pp. 1-5 (2013) doi: 10.1109/EWDTS.2013.6673183
12. Coppad, D., Sokolov, D., Bystrov, A., Yakovlev, A.: Online Testing by Protocol Decomposition. In: IOLTS - 12th IEEE International On-Line Testing Symposium, Como, Italy, pp. 263-268 (2006) doi: 10.1109/IOLTS.2006.45
13. Drozd, A., Drozd, J., Antoshchuk, S., Nikul, V., Al-dhabi, M.: Objects and Methods of On-Line Testing: Main Requirements and Perspectives of Development. In: EWDTS 2016 - IEEE East-West Design & Test Symposium, Yerevan, Armenia, pp. 72-76 (2016) doi: 10.1109/EWDTS.2016.7807750
14. Anderson, D. A., Metze, G.: Design of Totally Self-Checking Circuits for n-out-of-m Codes. IEEE Trans. on Computers, vol. C-22, 263-269 (1973) doi: 10.1109/T-C.1973.223705
15. Metra, C., Schiano, L., Favalli, M., Ricco, B.: Self-checking scheme for the on-line testing of power supply noise. In: DATE 2002 - Design, Automation and Test in Europe Conference, Paris, France, pp. 832-836 (2002) doi: 10.1109/DATE.2002.998395
16. Chakrabarty, K., Swaminathan S.: Built-in self-testing of high-performance circuits using twisted-ring counters. In: ISCAS 2000 - IEEE International Symposium on Circuits and Systems, Geneva, Switzerland (2000) doi: 10.1109/ISCAS.2000.857029
17. 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
18. Drozd, A., Kharchenko, V., Antoshchuk, S., Sulima, J., Drozd, M.: Checkability of the digital components in safety-critical systems: problems and solutions. In: EWDTS 2011 - IEEE East-West Design & Test Symposium, Sevastopol, Ukraine, 2011, pp. 411-416 doi: 10.1109/EWDTS.2011.6116606
19. 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, Studies in Systems, Decision and Control, vol. 171, Berlin, Heidelberg: Springer International Publishing, pp. 73-94 (2019) doi: 10.1007/978-3-030-00253-4_4
20. Gray, R.: The true toll of the Chernobyl disaster, BBC Future (2019) https://www.bbc.com/future/article/20190725-will-we-ever-know-chernobyls-true-death-toll
21. Gillis, D.: The Apocalypses that Might Have Been. [Online]. Available: https://www.damninteresting.com/the-apocalypses-that-might-have-been/.
22. Edstrom, J., Tilevich, E.: Reusable and Extensible Fault Tolerance for RESTful Applications. In: 11th International Conference on Trust, Security and Privacy in Computing and Communications, Liverpool, UK, pp. 737-744 (2012) doi: 10.1109/TrustCom.2012.244
23. Atamanyuk, I., Kondratenko, Y.: Computer's Analysis Method and Reliability Assessment of Fault-Tolerance Operation of Information Systems. In: CEUR-WS, vol. 1356, Lviv, Ukraine, pp. 507-522 (2015)
24. Romankevich, A., Feseniuk, A., Maidaniuk, I., Romankevich, V.: Fault-tolerant multiprocessor systems reliability estimation using statistical experiments with GL-models. In: Advances in Intelligent Systems and Computing, vol. 754, pp. 186-193 (2019)

25. Drozd, J., Drozd, A., Al-dhabi, M.: A resource approach to on-line testing of computing circuits. In: EWDTS 2015 - IEEE East-West Design & Test Symposium, Batumi, Georgia, pp. 276-281 (2015) doi: 10.1109/EWDTS.2015.7493122
26. Drozd, O., Kharchenko, V., Rucinski, A. et. al.: Development of Models in Resilient Computing. In: DESSERT 2019 - 10th IEEE International Conference on Dependable Systems, Services and Technologies, Leeds, UK, pp. 2-7 (2019) doi: 10.1109/DESSERT.2019.8770035
27. Tyurin, S.F., Grekov, A.V., Gromov, O.A.: The principle of recovery logic FPGA for critical applications by adapting to failures of logic elements. World Applied Sciences Journal, 328-332 (2013) doi: 10.5829/idosi.wasj.2013.26.03.13474
28. Jaecheon Jung, Ibrahim Ahmed: Development of FPGA-based reactor trip functions using systems engineering approach, Nuclear Engineering and Technology, March 2016, pp. 2-11 (2016) doi: 10.1016/j.net.2016.02.011
29. Palagin, A., Opanasenko, V.: The implementation of extended arithmetic's on FPGA-based structures. In: IDAACS 2017 - 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, vol. 2, Bucharest, Romania, pp. 1014-1019 (2017) doi: 10.1109/IDAACS.2017.8095239
30. 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) doi: https://doi.org/10.15587/1729-4061.2018.133405
31. 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)
32. Cyclone II Architecture. Cyclone II Device Handbook Version 3.1.-Altera Corporation (2007) http://www.altera.com/literature/hb/cyc2/cyc2_cii51002.pdf
33. Toshinori, S.: Basic Knowledge to Understand FPGAs. In: Principles and Structures of FPGAs, H. Amano (edit), Springer, USA, New-York, pp. 1-22 (2018)
34. Zashcholkin. K., Ivanova, O.: LUT-object integrity monitoring methods based on low impact embedding of digital watermark. In: TCSET 2018 - International Conference "Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering, Lviv-Slavske, Ukraine, pp. 519-523 (2018) doi: 10.1109/TCSET.2018.8336255
35. Drozd, A., Drozd, M., Kuznietsov, M.: Use of Natural LUT Redundancy to Improve Trustworthiness of FPGA Design. In: CEUR Workshop Proceedings, vol. 1614, pp. 322-331 (2016)
36. Pleskacz, W., Jenihhin, M., Raik, J. et. al.: Hierarchical Analysis of Short Defects between Metal Lines in CMOS IC. In: 11th Euromicro Conference on Digital System Design Architectures, Methods and Tools, Parma, Italy, pp. 729-734 (2008) doi: 10.1109/DSD.2008.98
37. Drozd, A., Drozd, M., Martynyuk, O., Kuznietsov, M.: Improving of a Circuit Checkability and Trustworthiness of Data Processing Results in LUT-based FPGA Components of Safety-Related Systems. In: CEUR Workshop Proceedings, vol. 1844, pp. 654-661 (2017)
38. Intel Quartus Prime Standard Edition User Guide: Getting Started, https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug-qps-getting-started.pdf, last accessed 2019/03/20
39. Max 10 FPGA Device Architecture (2017), https://www.intel.com /content/dam/www/programmable/us/en/pdfs/literature/hb/max-10/m10_architecture.pdf, last accessed 2019/03/20
40. Delphi 10 Seattle: Embarcadero (2015) https://www.embarcadero.com/ru/products/delphi/
41. Drozd, O., Kuznietsov, M., Martynyuk, O., Drozd, M.: A method of the hidden faults elimination in FPGA projects for the critical applications. In: DESSERT 2018 - 9th IEEE International Conference on Dependable Systems, Services and Technologies, Kyiv, Ukraine, pp. 231-234 (2018) doi: 10.1109/DESSERT.2018.8409131