DOI: https://doi.org/10.15276/aait.02.2021.4
UDC 004.315

IMPROVING FPGA COMPONENTS OF CRITICAL SYSTEMS BASED ON NATURAL VERSION REDUNDANCY

Oleksandr V. Drozd[1)]
ORCID: https://orcid.org/0000-0003-2191-6758; drozd@ukr.net
Andrzej Rucinski[2)]
ORCID: https://orcid.org/0000-0002-0988-7376; andrzej.rucinski@unh.edu
Kostiantyn V. Zashcholkin[1)]
ORCID: https://orcid.org/0000-0003-0427-9005; const-z@te.net.ua
Myroslav O. Drozd[1)]
ORCID: https://orcid.org/0000-0003-0770-6295; myroslav.drozd@opu.ua
Yulian Yu. Sulima[3)]
ORCID: https://orcid.org/0000-0003-3986-7296; mr_lemur@ukr.net
[1)] Odessa National Polytechnic University, 1, Shevchenko Avenue. Odesa, 65044, Ukraine
[2)]University of New Hampshire, Durham, New Hampshire 03824. Boston, USA
[3)]Odessa Technical Professional College of the Odessa National Academy of Food Technology, 54, Balkivska St. Odesa, 65006, Ukraine

ABSTRACT

The article is devoted to the problem of improving FPGA (Field Programmable Gate Array) components developed for safety-related systems. FPGA components are improved in the checkability of their circuits and the trustworthiness of the results calculated on them to support fault-tolerant solutions, which are basic in ensuring the functional safety of critical systems. Fault-tolerant solutions need protection from sources of multiple failures, which include hidden faults. They can be accumulated in significant quantities during a long normal operation and disrupt the functionality of fault-tolerant circuits with the onset of the most responsible emergency mode. Protection against hidden faults is ensured by the checkability of the circuits, which is aimed at the manifestation of faults and therefore must be supported in conjunction with the trustworthiness of the results, taking into account the decrease in trustworthiness in the event of the manifestation of faults. The problem of increasing the checkability of the FPGA component in normal operation and the trustworthiness of the results calculated in the emergency mode is solved by using the natural version redundancy inherent in the LUT-oriented architecture (Look-Up Table). This redundancy is manifested in the existence of many versions of the program code that preserve the functionality of the FPGA component with the same hardware implementation. The checkability of the FPGA component and the trustworthiness of the calculated results are considered taking into account the typical failures of the LUT-oriented architecture. These malfunctions are investigated from the standpoint of the consistency of their manifestation and masking, respectively, in normal and emergency modes on versions of the program code. Malfunctions are identified with bit distortion in the memory of the LUT units. Bits that are only observed in emergency mode are potentially dangerous because they can hide faults in normal mode. Moving potentially dangerous bits to checkable positions, observed in normal mode, is performed by choosing the appropriate versions of the program code and organizing the operation of the FPGA component on several versions. Experiments carried out with the FPGA component using the example of an iterative array multiplier of binary codes have shown the effectiveness of using the natural version redundancy of the LUT-oriented architecture to solve the problem of hidden faults.

Keywords: Safety-Related System; FPGA Component; LUT-Oriented Architecture; Functional Safety; Fault Tolerance; Checkability; Trustworthiness; Multiple Failures; Hidden Fault; Natural Version Redundancy; Versions of the Program Code

***For citation*: Drozd O. V., Rucinski Andrzej, Zashcholkin K. V., Drozd M. O., Sulima Yu. Yu. Improving FPGA Components of Critical Systems Based on Natural Version Redundancy. *Applied Aspects of Information Technology.* 2021; Vol. 4 No. 2: 168–177. DOI: https://doi.org/10.15276/aait.02.2021.4**

INTRODUCTION

FPGA-designing (Field Programmable Gate Array) has gained recognition in critical applications, where it is widely used in the development of safety-related systems.

According to international standards, these systems are aimed at ensuring the functional safety of high-risk facilities, which include power units of power plants and power grids, chemical production, transport infrastructures and much more [1, 2]. These objects have a twofold relationship to safety. On the one hand, they themselves ensure its various types of safety, including energy and food safety, as well as safety associated with the prompt delivery and distribution of produced and consumed resources.

On the other hand, the operation of these facilities is associated with an increased risk of accidents, which can lead to significant negative consequences. The potential cost of possible losses from accidents is constantly growing along with the increase in the number of facilities and their capacity.

In these conditions, the only way to contain risks is based on the development of information technologies implemented in computer systems for their transformation into safety-related systems.

A feature of these systems is their designing for operation in two modes: normal and emergency. The functional safety of the critical system is considered in conjunction with the safety of the high-risk object it controls.

The programmability of FPGAs with LUT-oriented architecture (Look-Up Table) creates the prerequisites for the development of multi-version technologies that are important for ensuring the functional safety of systems in critical domains.

For safety-related systems, the main focus in improving their FPGA components is to improve functional safety, which is based on the use of fault-tolerant solutions.

At the same time, the fault tolerance of the structures is ensured in relation to one or two failures, taking into account the balance between a significant decrease in the probability of independent failures with an increase in their multiplicity and the complication of the solution, leading to its rise in cost and losses in reliability [3, 4].

However, the independence of failures, as a rule, is of a relative nature and at one level or another show the common causes of their occurrence. In addition, the decisive factor is the manifestation of multiple failures, not their occurrence. Therefore, the greatest challenge to fault tolerance in ensuring the functional safety of safety-related systems is the sources of multiple failures.

An important place among such sources is occupied by the limited checkability of digital circuits, which normally operate in a limited range of input data. Insufficient checkability leads to the problem of hidden faults, which is inherent in modern safety-related systems at this stage of their development. Circuits can accumulate hidden faults during extended normal operation. In emergency mode, the manifestation of accumulated faults can significantly disrupt the fault tolerance of the circuits and reduce the trustworthiness of the calculated results.

Thus, the main indicators important for the functional safety of critical systems are the checkability of the schemes and the trustworthiness of the results calculated on these schemes. In fault-tolerant solutions, the functional safety of critical systems should be ensured by increasing the checkability of the circuits and the trustworthiness of the results, respectively, in the normal and emergency mode of FPGA components.

The goal of this paper is to improve FPGA components in their checkability, maintained during normal operation, and the trustworthiness of results calculated in emergency mode. The improvement is achieved by using the natural version redundancy of the LUT-oriented architecture.

Further presentation of the research carried out is organized as follows. Section 2 describes the prerequisites for the further development of FPGA designing towards improving circuit checkability and trustworthiness of results in critical applications. Section 3 reveals the possibilities of version redundancy of the LUT-oriented architecture to improve FPGA components in key indicators aimed at ensuring the functional safety of critical systems. Section 4 shows the results of experiments carried out with FPGAs to improve their basic performance.

BACKGROUND FOR FPGA DEVELOPMENT IN CRITICAL APPLICATIONS

The prerequisites for improving FPGA designing consist of the necessity and possibility of such development in safety-related systems.

For critical systems, the need to improve FPGA components is most clearly demonstrated by the lack of confidence in the applied fault-tolerant solutions. This mistrust is manifested in the use of simulation modes that recreate emergency conditions to increase the checkability of circuits and detect hidden faults in them.

However, simulation modes themselves pose a great danger to safety-related systems and objects of their control. The very existence of these modes carries a certain threat, since it has repeatedly led to their unauthorized activation due to operator error and even one burned-out microcircuit. The planned use of simulation modes is associated with the shutdown of emergency protection, which served as one of the causes of the Chernobyl accident [5, 6].

For safety-related systems, the development paths of components are determined in the resource-

based approach, which proposes to develop models, methods and tools that form resources, analyzing the integration of the computer world, created by human, into the natural world. In resource development, this approach identifies three levels: replication, diversification and self-sufficiency as a development goal [7].

Replication is the lowest level of development, at which integration occurs at the expense of productivity in open resource niches, i.e., in the absence of rigid contact with the natural world. The computer world demonstrates all levels of development, but replication prevails.

The hardware is represented by matrix circuits stamped out of the same operating elements: parallel adders, shifters, registers, iterative array multipliers and dividers.

The software is compiled by replicating ready-made modules, which can be functionally redundant to a large extent for a given application and, therefore, have unreasonably large sizes. However, this replication is supported by open resource niches in the throughput and memory size of modern computers.

Mobile computer systems are limited by the resource niche of autonomous energy consumption and require the development of green technologies, which belong to the next level of diversification [8, 9].

To understand the inferiority of the lower level of development, it is enough to consider a bright pattern of matrix structures using the example of an iterative array multiplier that performs an operation in one clock cycle. Its circuit consists of n^2 operational elements, $2n-2$ of which are connected in series, where n is the size of the operands. Therefore, each operational element is used only for a small part of $1 / (2n-2)$ clock cycle, which is 0.8% for $n = 64$ [10].

The rest of the time is spent on parasitic switching caused by signal races. They determine the main part of the dynamic component of energy consumption, as well as the static component is determined by the large size of the matrices [11, 12].

However, the greatest disadvantage of matrix structures is their low checkability, which is due to the processing of data in parallel codes. For $n = 64$, the iterative array multiplier input word contains 128 bits and can take 2^{128} different values, of which normal mode can only use a few values.

The resource-based approach defines the problem of hidden faults as a growth challenge, when the safety-related system rises to the level of diversification, which manifests itself in the operating mode by its division into normal and emergency, in the input data of the digital circuit and the checkability dependent on them. They all become different in these modes. At the same time, the components of these systems continue to be stamped at the replication level in the form of matrix structures.

It should be noted that the problem of hidden faults does not arise in conventional computers working in only one operating mode. Both the system and its components are at the same level of development, and retain the hidden nature of the malfunction throughout the entire operating mode.

This understanding of the problem opens the way for its solution: the components must be raised to the level of the system. It should be borne in mind that matrix structures have dominated for several decades and during this time they have created powerful support resources, including CAD. FPGA designing offers on-chip iterative array multiplier circuits, pre-built carry propagation paths for accelerated data addition in parallel codes, and a wide range of library solutions based on matrix structures.

Therefore, today, raising components to the level of diversification should be done in matrix structures, including the LUT-oriented structure of FPGA components designed for critical applications.

It should be noted that in conventional computers checkability and trustworthiness are in some opposition. The higher the checkability of the circuit, the better the malfunctions will manifest, including the faults that are most likely to occur. In this case, erroneous results are calculated more often and their trustworthiness decreases.

Developing resources towards self-sufficiency removes inconsistencies, including the performance of FPGA components. For safety-related systems, the requirements for improving the checkability of the schemes and the trustworthiness of the results are spread by their application to different modes: normal and emergency, respectively.

Some experience in resisting multiple failures has already been accumulated in international standards in relation to common cause failures [13, 14]. This reason is seen in the copying of erroneous decisions that can arise, for example, as a result of design errors. Installing the same software on duplicate

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)

components can cause multiple failures in the event of an error in the program.

The standards recommend limiting the common cause by using multi-version technologies [15], reflecting the level of diversification. Development of versions of the program by different teams of programmers significantly reduces the likelihood of identical errors.

The LUT-oriented architecture of FPGA components organizes the execution of computations by decomposing them into logical functions of several arguments. The generator of these functions is the LUT node, which, as part of the logical element LE, is supplemented with a programmable one-bit register. The LUT node is a SRAM memory for storing a description of a function, which is written into this memory in the form of program code during the programming of the FPGA component. In the widespread case of decomposition of computations into functions for no more than four arguments, the LUT node has 16 bits of memory, addressed by the $dcba_2$ code, which is fed to 4 inputs D, C, B, A [16].

Version redundancy of the LUT-oriented architecture consists in the presence of many versions of the program code for the same hardware implementation of the FPGA component.

The forming element of version redundancy is a pair of LUT nodes connected in series, i.e., the output of the first node is connected to any input of the second node of the pair. In this case, the original version of the program code can be supplemented with another version, which differs in the inversion of the bit transmitted from the first LUT node to the second node of the pair. Bit inversion at the output of the first LUT node can be provided by inverting the program code in bits related to the memory of this node. The bit inversion at the input of the second node of the pair is compensated by renumbering the bits of its memory in the program code of the FPGA component [17].

The number of pairs is determined by the number of the first LUT nodes in the FPGA project. Branching the circuit at the output of the first LUT node does not increase the number of pairs that create versions of the program code. All these versions are created independently of each other and therefore ensure their growth as a power function of the number of pairs.

For example, one hardware implementation of an FPGA project will have over a million versions of the program code with only twenty pairs of LUT nodes. This number of pairs is formed in a pyramid scheme that contains 21 LUT nodes located in three tiers of 16, 4 and 1. Only a single LUT node in the third tier is not the first node in the pair.

It should be emphasized the natural kind of the described version redundancy, which does not need to be introduced into the FPGA projects, since it is an integral part of the LUT-oriented architecture of this design.

Thus, FPGA components have significant natural version redundancy even with insignificant complexity determined by the number of LUT nodes.

It should be noted that the indicators characterizing the FPGA component can be divided in relation to versions into two groups: active and passive. Versions differ in active metrics and are equal in passive metrics. Version redundancy plays an important positive role for both groups of metrics because versions with the best active metrics can be selected without loss of passive metrics.

In the studies carried out, the active indicators are the checkability of the digital circuit and the trustworthiness of the results calculated on it. The preservation of the hardware implementation of the FPGA component refers to the passive indicators of the complexity of the circuit design, its reliability characteristics and, possibly, many other indicators.

For example, power consumption also cannot undergo any changes in its static and dynamic components, taking into account the preservation of the circuit and the number of signal transition at the inputs / outputs of the LUT nodes, respectively.

IMPROVING THE CHECKABILITY OF SCHEMES AND TRUSTWORTHINESS OF RESULTS

Both checkability and trusyworthiness are associated with circuit malfunctions and their manifestation in the form of a result error or masking of this manifestation. Therefore, both indicators must be considered in relation to the characteristic faults inherent in circuits with a LUT-oriented architecture.

A fault of the LUT node can manifest itself in the form of erroneous bit values at its inputs or outputs as a result of short-circuiting adjacent inputs or reading from memory.

The program code written to the nodes LUT memory from the configuration file is checked with a checksum. Therefore, the nodes' LUT memory is

checkable. The bits of this memory are read using a multiplexer consisting of switches that select a bit from two directions in one direction under the control of the address bits $dcba_2$ [18].

A malfunction of the switches at their information inputs can cause an error if the correct bit value and its value determined by the malfunction do not match. A malfunction of the control input of the switch leads to an addressing error, which distorts the result at the output of the LUT node, if the memory bits at the correct and erroneous address do not match in their values. Errors caused by faults in the switches are perceived at the output of the LUT node as a distortion of the corresponding memory bits, and therefore their checkability and the trustworthiness of the results are further considered with respect to these bits and their constant malfunction in the memory of the LUT node.

The checkability of the circuit and the trustworthiness of the results with respect to short circuit faults occurring between adjacent inputs of the LUT nodes should be considered taking into account the peculiarities of the manifestation and masking of these faults.

A short circuit fault manifests itself as an error in the $dcba_2$ address code if the inputs to be closed should have taken different values and is masked otherwise. At the same time, an error at the output of the LUT node appears if the bits read at the correct and erroneous addresses differ in the values they receive. Otherwise, this addressing error is masked.

The manifestation of a short circuit fault at the output of the LUT node classifies it as checkable if this error is transmitted to the monitored result of calculations, i.e., if the node's LUT output is observable. Otherwise, the error does not reach the result, and this result is considered reliable [19].

An important consideration is the diversification of the data set at inputs of the LUT node into two non-overlapping subsets that manifest or mask a short circuit fault. Indeed, the values 01_2 and 10_2 at the adjacent inputs of the LUT node exhibit a short circuit fault, while the values 00_2 and 11_2, on the contrary, mask it.

This diversification creates favorable conditions for finding versions of the program code with the diversity of the best checkability and trustworthiness in different modes of the critical system in accordance with different input data of these modes. Inversion at the input of the second LUT node of the pair allows to swap the indicated subsets in the manifestation or masking of a short circuit fault. This capability provides the selection of a version with fault manifestation and masking, i.e., the best checkability and trustworthiness for the input data of the normal and emergency mode, respectively.

Erroneous reading of bits from the node's LUT memory is best distributed with respect to the checkability and trustworthiness indicators with a dominant constant fault, i.e., the dominance of a constant of one value over the opposite. In this case, the memory bits, the value of which is inverse to the value of the dominant constant, belong to the checkable at the output of the LUT node, since the action of the corresponding malfunction leads to bit corruption. This error will be detected if the node's LUT is monitored. Otherwise, the fault determines the correct value of the memory bit, and the action of the fault does not reduce the trustworthiness of the calculated result.

For safety-related systems, fault detection should be performed prior to failure by ensuring the necessary checkability of the LUT-oriented architecture in normal operation. With the beginning of the emergency mode, the manifestation of malfunctions should be maximally excluded by masking them in order to calculate reliable results. This problem is solved by generating versions of the program code and choosing the versions with the best indicators of the checkability of the circuit and the trustworthiness of the results, respectively, in normal and emergency modes.

The best solution is provided by versions that completely eliminate the occurrence of faults in emergency mode and thus maintain the required trustworthiness of the results. In the absence of such versions, preference is given to versions with maximum fault masking in emergency mode, provided that the circuit is checkable for other faults. This case also includes the absence of masked faults if they all manifest in normal mode.

All described versions of the program code completely solve the problem of hidden faults by leveling the checkability of the normal mode to the level of the emergency mode. In this case, the malfunction hidden in the normal mode retains this state in the emergency mode, and the remaining malfunctions lose their hidden character and therefore will be detected in the normal mode by means of on-line testing.

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)

If none of the versions provides a complete solution to the problem of hidden faults, then such a solution can be obtained by organizing the operation of the FPGA component on several versions of the program code, changing sequentially in time. Thus, it is possible to combine the use of versions that successfully withstand several types of faults, including the dominance of opposite stuck-at faults in the memory of the LUT nodes. Fault resistance is carried out by improving the checkability of the circuit in normal mode, since it is impossible to predict which version the emergency mode will start on.

Trustworthiness must be ensured for the memory bits observed in emergency mode, since only these bits affect the computed results. Some of these bits can be observed in normal mode, i.e., be checkable. The rest of the bits, observed only in emergency mode, are potentially dangerous, since they create the problem of hidden faults. This problem can be solved by moving potentially dangerous bits to checkable positions.

The numbering of the memory bits by the value of the $dcba_2$ address, and the versions by a binary code, the bits of which take one value in the case of inversion of the corresponding input of the LUT node, allows to determine the version for moving a memory bit from one position to another according to the formula: $V_{KL} = N_K \oplus N_L$, where V_{KL}, N_K, N_L – version numbers and moved bits, respectively [17].

It should also be noted the dependence of the checkability of the circuit and the trustworthiness of the results on the ranges of the input data in normal and emergency modes. As the range of inputs used in normal mode increases, the checkability of the schema increases, reducing the requirements to selection of versions. Reducing the range of the emergency mode helps masking faults and increasing the trustworthiness of the results calculated in this mode. This circumstance should be taken into account when it is possible to influence the ranges of the input data of normal and emergency modes, for example, in the process of forming specifications for the design of an FPGA component.

EXPERIMENTAL STUDIES

The experiments are aimed at demonstrating the described approaches to improving FPGA components in terms of checkability and trustworthiness by using the natural version redundancy of the LUT-oriented architecture.

The FPGA component under study implements an iterative array multiplier of 8-bit binary codes taken from the library LPM_mult CAD Quartus [20]. The multiplier is implemented in Intel Cyclone 10 LP FPGA chip: 10CL025YU256I7G [21] using CAD Quartus Prime 20.1 Lite Edition [22].

The FPGA component circuit contains 101 LUT nodes and forms 85 pairs of them, creating 285 versions of the program code.

The description of the FPGA component contains the numbers of the LUT nodes with an indication of their program code and connection to other LUT nodes, as well as the inputs and outputs of the circuit.

Versions of the program code were obtained using a program developed in the Delphi 10 Seattle demo version [23].

The program uses the description of the FPGA component to simulate the process of performing computations in accordance with the LUT-oriented iterative array multiplier circuit. For each LUT node, the simulation determines the observability of the memory bits in normal and emergency mode.

The division of the input data into normal and emergency ranges is done using the threshold *S*. Factors less than *S* determine input data of the normal mode. When at least one factor reaches or exceeds the threshold, the input data refers to the emergency mode.

The program performs 4 experiments for different values of the threshold *S*.

Code versions are generated independently for each pair of LUT nodes.

Fig. 1 shows the main panel of the program for modeling the LUT-oriented circuit of the iterative array multiplier.

The panel shows the control keys for starting the simulation and exiting the program, as well as setting the *S* threshold and the LUT number of the node, the memory of which is shown below for four *S* threshold values. The threshold is set in the range from 2 to 50 and changes in increments of 16.

A node's LUT memory is shown as a matrix of bits. Its rows and columns are numbered from 00_2 to 11_2 with the values of the bits arriving at the D, C and B, A inputs of the LUT node, respectively. Bits located at the intersection of rows and columns are numbered with the $dcba_2$ address. For example, the upper right and lower left bits are addressed with codes 0011_2 and 1100_2, respectively.

Checkable bits, observed in normal mode, and potentially dangerous bits, observed only in emergency mode, are colored blue and yellow, respectively. The values of potentially dangerous bits are highlighted in blue if the versions of the program code for moving these bits are generated only when the circuit inputs are inverted.

The number of versions required to move potentially dangerous bits to checkable positions and the numbers of those versions starting from the original zero version are shown below the memory matrices. Version numbers are represented in hexadecimal digits.

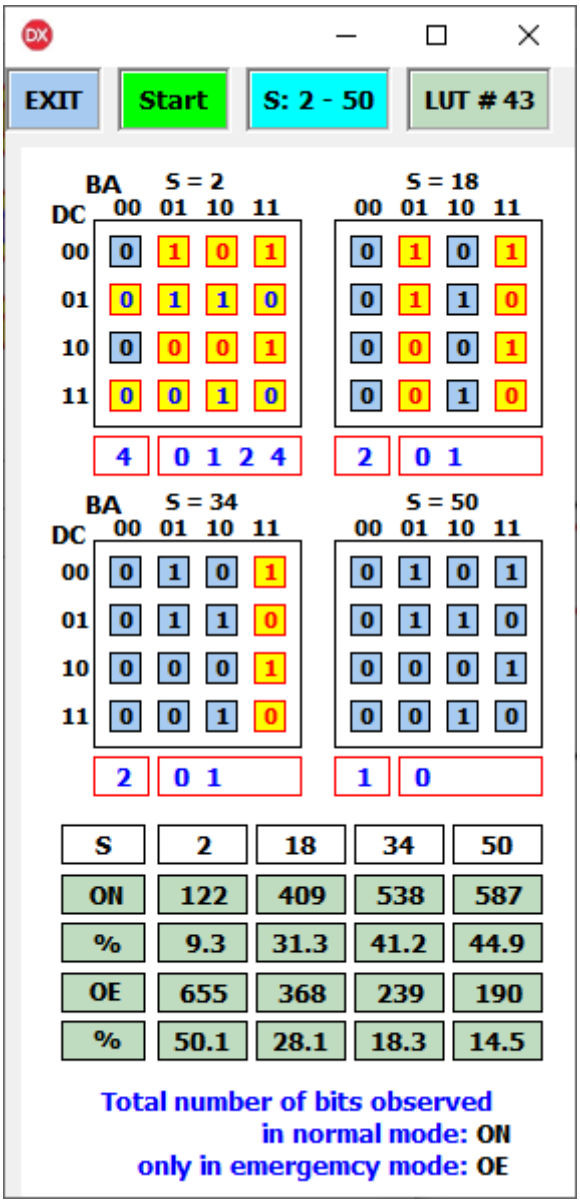

Fig. 1. **The main panel of the iterative array multiplier simulator**

Source: **compiled by the authors**

Memory and versions are shown using the example of LUT node 43. In the case of $S = 2$, bits 0000_2 and 1000_2 are checkable. The rest of the bits are potentially dangerous.

Bits 0100_2 – 0111_2 and 1100_2 – 1111_2 can be moved to checkable positions by inverting the inputs of the schema. There are 4 versions used to move the bits. Version 1_{16} swaps the bits of the first and second and third and fourth columns of the matrix and adds bits 0001_2 and 1001_2 to the checkable bits. Version 2_{16} swaps the left and right halves of the matrix by attaching bits 0010_2, 0011_2, and 1010_2 1011_2 to the checkable bits. Version 4_{16} swaps the bits of the first and second, as well as the third and fourth rows of the matrix, and thus moves the remaining potentially dangerous bits to checkable positions.

In the case of $S = 18$ and $S = 34$, all potentially dangerous bits are moved to checkable positions using one version 1_{16}.

The threshold $S = 50$ turns out to be high enough to ensure checkability of the entire memory of LUT node 43 on the original version of the program code.

Raising the S threshold demonstrates a change in the color of the memory bits from the dominant yellow to blue. Potentially dangerous bits are transformed into checkable bits. This trend can also be seen in the simulation results shown at the bottom of the main program panel. As the threshold increases, the number of checkable bits observed in normal mode increases from 122 (9.3 %) to 587 (44.9%). At the same time, the number of potentially dangerous bits decreases from 655 (50.1%) to 190 (14.5 %).

CONCLUSION

Ensuring the functional safety of critical systems is based on the use of fault-tolerant solutions, the effectiveness of which is significantly limited by sources of multiple failures, including hidden faults arising from insufficient checkability of digital circuits. Therefore, FPGA components of safety-related systems need to improve their checkability, which is maintained during normal operation in order to increase the trustworthiness of the results calculated in emergency mode.

The proposed improvement of FPGA components is based on the use of the natural version redundancy of the LUT-oriented architecture, which manifests itself in different versions of the program

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)

code that exist for the same hardware implementation of the component.

Analysis of typical failures of LUT nodes showed natural opportunities for their manifestation and, on the contrary, masking, respectively, in normal and emergency modes in order to combine high checkability of the circuit and the trustworthiness of the results achieved in versions of the program code.

The experiments have shown the possibility of solving the problem of hidden faults by organizing the operation of the FPGA component on several versions of the program code, which ensure the movement of potentially dangerous bits of the LUT memory, observed only in emergency mode, to checkable positions that are observed during the normal operation of the safety-related system.

REFERENCES

1. Smith, D. & Simpson, K. "The Safety Critical Systems Handbook". [5th ed.]. Butterworth-Heinemann: 2019. DOI: https://doi.org/10.1016/C2019-0-00966-1.

2. Otradskaya, T. V., Rudnichenko, N. M., Shibaev, D. S., Shibaeva, N. O. & Vychuzhanin, V. V. "Data Control in the Diagnostics and Forecasting the State of Complex Technical Systems". *Herald of Advanced Information Technology. Publ. Nauka i Tekhnika*. Odesa: Ukraine. 2019; Vol. 2 No. 3: 183–196. DOI: https://doi.org/10.15276/hait.03.2019.2.

3. Tyurin, S. F. "Investigation of a Hybrid Redundancy in the Fault-Tolerant Systems". *Radio Electronics, Computer Science, Control.* 2019; Vol. 2: 23–33. DOI: https://doi.org/10.15588/1607-3274-2019-2-3.

4. Arya, N. & Singh, A. P. "Fault Tolerant System for Embedded System Architecture". *International Journal of Engineering and Technology (IJET)*. 2017; Vol.9 No.3: 93–97. DOI: https://doi.org/10.21817/ijet/2017/v9i3/170903S016.

5. Gillis, D. "The Apocalypses that Might Have Been". – Available from: https://www.damninteresting.com/the-apocalypses-that-might-have-been/. 2007. – [Accessed 20th Mar. 2019].

6. Hussain, Y., Rehalia, A. & Dhyan, A. "Case Study: Chernobyl Disaster". *International Journal of Advanced Research in Computer Science and Software Engineering*. 2018; Vol.8 No.2: 76–78.

7. Kovalev, I. S., Drozd, O. V., Rucinski, A., Drozd, M. O., Antoniuk, V. V. & Sulima, Y. Y. "Development of Computer System Components in Critical Applications: Problems, Their Origins and Solutions". *Herald of Advanced Information Technology. Publ. Nauka i Tekhnika*. Odesa: Ukraine. 2020; Vol.3 No.4: 252–262. DOI: https://doi.org/10.15276/hait.04.2020.4.

8. Murugesan, S. & Gangadharan, G. "Harnessing Green IT". Principles and Practices; Wiley and Sons Ltd.: Hoboken. NJ: USA. 2012.

9. Kharchenko, V., Gorbenko, A., Sklyar, V. & Phillips, C. "Green Computing and Communications in Critical Application Domains: Challenges and Solutions". *In: IX International Conference of Digital Technology.* Zhilina: Slovak Republic. 2013. DOI: https://doi.org/10.1109/DT.2013.6566310.

10. Drozd, J., Drozd, A., Antoshchuk, S., Kushnerov, A. & Nikul, V. "Effectiveness of Matrix and Pipeline FPGA-Based Arithmetic Components of Safety-Related Systems". *The 8th IEEE International Conference IDAACS.* Warsaw: Poland. 2015. p. 785–789. DOI: https://doi.org/10.1109/IDAACS.2015.7341410.

11.Warren, S. & Anderson, J. "FPGA Glitch Power Analysis and Reduction". *In: International Symposium on Low Power Electronics and Design.* Fukuoka: Japan. 2011. p. 27–32. DOI: https://doi.org/10.1109/ISLPED.2011.5993599.

12.Velegalati, R. & Kaps, J.-P. "Glitch Detection in Hardware Implementations on FPGAs Using Delay Based Sampling Techniques". *In: Euromicro Conference on Digital System*. Design Los Alamitos. CA: USA. 2013. DOI: https://doi.org/10.1109/DSD.2013.107.

13. Alizadeh, S. & Sriramula, S. "Impact of Common Cause Failure on Reliability Performance of Redundant Safety Related Systems Subject to Process Demand". *Reliability Engineering & System Safety*. 2018; Vol. 172: 129–150. DOI: https://doi.org/10.1016/j.ress.2017.12.011.

14. Kumar, M., Kabra, A., Karmakar, G. & Marathe, P. P. "A Review of Defences against Common Cause Failures in Reactor Protection Systems". *In: 4th International Conference on Reliability, Infocom*

Technology and Optimization (ICRITO). Noida: India. 2015. p. 1–5. DOI: https://doi.org/10.1109/ICRITO.2015.7359232.

15. Kharchenko, V., Bakhmach, E., Siora, A., Sklyar, V. & Tokarev, V. "Diversity-Oriented FPGA-Based NPP I&C Systems: Safety Assessment, Development and Implementation". *18th International Conference on Nuclear Engineering.* Xi'an: China. 2010. p. 755–764. DOI: https://doi.org/10.1115/ICONE18-29754.

16. Ebrahimi, M., Sadeghi, R. & Navabi, Z. "LUT Input Reordering to Reduce Aging Impact on FPGA LUTs". *IEEE Transactions on Computers.* 2020; Vol.69 No.10: 1500–1506. DOI: https://doi.org/10.1109/TC.2020.2974955.

17. Drozd, O., Zashcholkin, K., Martynyuk, O., Ivanova, O. & Drozd J. "Development of Checkability in FPGA Components of Safety-Related Systems". *CEUR Workshop Proceedings*. 2020; Vol. 2762: 30–42. Available from: http://ceur-ws.org/Vol-2762/paper1.pdf.

18. Amano, H. "Principles and Structures of FPGAs". *Publ. Springer.* Singapore: 2018. DOI: https://doi.org/10.1007/978-981-13-0824-6.

19. Shah, T., Matrosova, A., Fujita, M. et. al. "Multiple Stuck-at Fault Testability Analysis of ROBDD Based Combinational Circuit Design". *Journal of Electronic Testing*. 2018; Vol.34 No.1: 53–65. DOI: https://doi.org/10.1007/s10836-018-5703-3.

20. "Intel FPGA Integer Arithmetic IP Cores User Guide". 2020. – Available from: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug_lpm_alt_mfug.pdf.

21. "Intel Cyclone 10 LP Core Fabric and General Purpose I/Os Handbook". 2020. – Available from: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/cyclone-10/c10lp-51003.pdf.

22. "Intel Quartus Prime Standard Edition User Guide". 2020. – Available from: https://www.intel.com/content/dam/ altera-www/global/en_US/pdfs/literature/ug/ug-qps-getting-started.pdf.

23. "Delphi 10 Seattle: Embarcadero". – Available from: https://www.embarcadero.com/docs/datasheet.pdf.

Conflicts of Interest: the authors declare no conflict of interest

Received 23.12.2020
Received after revision 11.03.2021
Accepted 17.03.2021

DOI: https://doi.org/10.15276/aait.02.2021.4
УДК 004.315

ПОКРАЩЕННЯ FPGA-КОМПОНЕНТІВ КРИТИЧНИХ СИСТЕМ НА ОСНОВІ ПРИРОДНОЇ ВЕРСІЙНОЇ НАДМІРНОСТІ

Олександр Валентинович Дрозд[1)]
ORCID: https://orcid.org/0000-0003-2191-6758; drozd@ukr.net
Анджей Русінський[2)]
ORCID: https://orcid.org/0000-0002-0988-7376; andrzej.rucinski@unh.edu
Костянтин Вячеславович Защолкін[1)]
ORCID: https://orcid.org/0000-0003-0427-9005; const-z@te.net.ua
Мирослав Олександрович Дрозд [1)]
ORCID: https://orcid.org/0000-0003-0770-6295; myroslav.drozd@opu.ua
Юліан Юрійович Суліма[3)]
ORCID: https://orcid.org/0000-0003-3986-7296; mr_lemur@ukr.net
[1)] Одеський національний політехнічний університет, проспект Шевченка, 1. Одеса, 65044, Україна
[2)] Університет Нью-Гэмпшира, Дарем, Нью-Гэмпшир 03824. Бостон, США
[3)]Одеський технічний фаховий коледж Одеської національної академії харчових технологій, Балківська вул., 54. Одеса, 65006, Україна

АНОТАЦІЯ

Статтю присвячено проблемі вдосконалення FPGA-компонентів, що розробляються для систем критичного застосування. FPGA-компоненти поліпшуються в контролепридатності їх схем і достовірності обчислюваних на них результатів

ISSN 2617-4316 (Print)
ISSN 2663-7723 (Online)

для підтримки відмовостійких рішень, які є базовими в забезпеченні функціональної безпеки критичних систем. Відмовостійкі рішення потребують захисту від джерел кратних відмов, до яких відносяться приховані несправності. Вони можуть накопичуватися в значній кількості на протязі тривалого нормального режиму і порушувати функціональність відмовостійких схем з початком найбільш відповідального аварійного режиму. Захист від прихованих несправностей забезпечується контролепридатністю схем, яка націлена на прояв несправностей і тому повинна підтримуватися в комплексі з достовірністю результатів, беручи до уваги зниження достовірності при прояві несправностей. Завдання підвищення контролепридатності FPGA-компонента в нормальному режимі і достовірності результатів, що обчислюються в аварійному режимі, вирішується шляхом використання природної версійної надмірності, властивої LUT-орієнтованій архітектурі. Ця надмірність проявляється в існуванні множини версій програмного коду, що зберігають функціональність FPGA-компонента при одній і тій же його апаратної реалізації. Контролепридатність FPGA-компонента і достовірність обчислюваних результатів розглядаються з урахуванням характерних несправностей LUT-орієнтованої архітектури. Ці несправності досліджені з позиції несуперечності їх прояву і маскування відповідно в нормальному і аварійному режимі на версіях програмного коду. Несправності ототожнюються зі спотворенням бітів в пам'яті LUT вузлів. Біти, що спостерігаються тільки в аварійному режимі, є потенційно небезпечними, оскільки можуть приховувати несправності в нормальному режимі. Переміщення потенційно небезпечних бітів на контролепридатні позиції, які спостерігаються в нормальному режимі, виконується шляхом вибору відповідних версій програмного коду і організації роботи FPGA-компонента на декількох версіях. Експерименти, що проведені з FPGA-компонентом на прикладі матричного помножувача двійкових кодів, показали ефективність використання природної версійнної надмірності LUT-орієнтованої архітектури для вирішення проблеми прихованих несправностей.

Ключові слова: Система критичного застосування; FPGA-компонент; LUT-орієнтована архітектура; функціональна безпека; відмовостійкість; контролепридатність; достовірність; кратні відмови; прихована несправність; природна версійна надмірність; версії програмного коду

ABOUT THE AUTHORS

Oleksandr V. Drozd – Dr. Sci. (Eng.) (2003), Prof. of Computer Intellectual Systems and Networks Department. Odessa National Polytechnic University, 1, Shevchenko Avenue. Odesa, 65044, Ukraine
ORCID: https://orcid.org/0000-0003-2191-6758; drozd@ukr.net
***Research field*:** On-Line Testing; Green Technologies and Circuit Checkability in the Digital Component of Safety-Related Systems; LUT-Oriented Architecture of FPGA-Based Systems

Олександр Валентинович Дрозд – доктор технічних наук (2003), професор кафедри Комп'ютерних інтелектуальних систем та мереж. Одеський національний політехнічний університет, пр. Шевченка, 1. Одеса, 65044, Україна

Andrzej Rucinski – PhD Professor Emeritus, Department of Electrical and Computer Engineering, University of New Hampshire, a member of the Executive Committee (Innovation Chair) of the IEEE Computer Society's Design Automation Technical Committee. USA Ambassador of International Society of Service Innovation Professionals. University of New Hampshire, Durham, New Hampshire 03824. Boston, USA
ORCID: https://orcid.org/0000-0002-0988-7376; andrzej.rucinski@unh.edu
Research *field*: Ecosystem Grand Challenges Associated with Ehealth/Mhealth; Eeducation/Elearning; Esecurity/Identity Protection; Smart City/Region/State and Information Infrastructure Technology Involving a Digital Ecosystem Using Internet of Things

Анджей Русінський – почесний доктор філософії кафедра Електротехніки та обчислювальної техніки. Університет Нью-Гэмпшира, член Виконавчого комітету (голова з інновацій) Технічного комітету з автоматизації проєктування IEEE Комп'ютерного товариства, посол Міжнародного товариства професіоналів у області сервіс-інновацій. Університет Нью-Гэмпшира, Дарем, Нью-Гэмпшир 03824. Бостон, США.

Kostiantyn V. Zashcholkin – Dr. Sci. (Eng) (2020), Associate Professor of the Department of Computer Intelligent Systems and Networks. Odessa National Polytechnic University, 1, Shevchenko Avenue. Odessa, 65044, Ukraine
ORCID: https://orcid.org/0000-0003-0427-9005; const-z@te.net.ua
Research *field*: FPGA-Based Systems; Digital Watermarking; Digital Steganography

Костянтин Вячеславович Защолкін – доктор технічних наук (2020), доцент кафедри Комп'ютерних інтелектуальних систем та мереж. Одеський національний політехнічний університет, проспект Шевченка, 1. Одеса, 65044, Україна

Myroslav O. Drozd – PhD (2014), Associated Prof. of Information Systems Department. Odessa National Polytechnic University, 1, Shevchenko Avenue. Odessa, 65044, Ukraine
ORCID: https://orcid.org/0000-0003-0770-6295; myroslav.drozd@opu.ua
Research *field*: On-Line Testing and Circuit Checkability in the Digital Component of Safety-Related Systems

Мирослав Олександрович Дрозд – кандидат технічних наук (2014), доцент кафедри Інформаційних систем. Одеський національний політехнічний університет, проспект Шевченка, 1. Одеса, 65044, Україна

Yulian Yu. Sulima – PhD (2014), Head of the Computer Systems Department, SSU "Odessa Technical Professional College of the Odessa National Academy of Food Technology", 54, Balkivska St. Odessa, 65006, Ukraine
ORCID: https://orcid.org/0000-0003-3986-7296; mr_lemur@ukr.net
Research *field*: Technology of Designing Computer Systems on FPGA; Computer Systems for Critical Application; Checkability and Detection of Hidden Faults of Integrated Circuits

Юліан Юрійович Суліма – кандидат технічних наук (2014), завідувач відділення Комп'ютерних систем ВСП «Одеський технічний фаховий коледж Одеської національної академії харчових технологій», Одеса, Україна.