Checkability Important for Fail-Safety of FPGA-based Components in Critical Systems

Oleksandr Drozd[a], Olena Ivanova[a], Kostiantyn Zashcholkin[a], Vitaliy Romankevich[b] and Julia Drozd[a]

[a] *Odessa National Polytechnic University, Ave. Shevchenko 1, Odesa, 65044, Ukraine*
[b] *National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Victory avenue, 37, Kyiv, Ukraine*

Abstract
The paper is devoted to the analysis of FPGA (Field Programmable Gate Array) components with LUT-oriented (Look-Up Table) architecture for safety-related systems that are aimed at ensuring the functional safety of high-risk facilities in conjunction with their own safety. Functional safety is based on the use of fault-tolerant solutions, for which multiple failures are the biggest challenge. One of the sources of such failures is associated with the problem of hidden faults that can accumulate in digital circuits during a long normal mode and manifest themselves in a decrease or loss of the fault tolerance of these circuits in the most responsible emergency mode. The accumulation of faults occurs in connection with the limited checkability of digital circuits in normal mode and due to the change in checkability with the beginning of the emergency mode. The lack of checkability of the FPGA components, which manifests itself in the memory of the LUT units, does not allow the fault-tolerant circuit to be transformed into a fail-safe one. A method for assessing the checkability of circuits with LUT-oriented architecture in the part that ensures the fail-safety of a fault-tolerant solution is proposed. Two sets of memory LUT bits are determined that are important to the circuit in terms of providing fail-safety and its violation, respectively. The program implementation of the method demonstrates its capabilities on the example of an iterative array multiplier implemented in an FPGA project.

Keywords
Safety-related system, FPGA component, LUT-oriented architecture, memory bits of LUT unit, hidden faults, checkability fail-safety, fault tolerance

1. Introduction

The current stage in the development of computer systems shows an increase in their importance, primarily in critical applications associated with the development and operation of high-risk facilities. These facilities are typical for the energy sector, including power plants and power grids. Energy consumption and demand is constantly growing, stimulating an increase in the number and capacity of power plants, as well as the development of power grids for the transportation and distribution of energy. Systems for the transport of goods and passengers are also becoming critical infrastructures due to the increase in freight traffic, speeds and ramifications of communication lines. The processes of property redistribution and their expectations stimulate the development of various types of weapons, which in their modern design are also classified as high-risk objects. Chemical and biological industries, as well as warehouses for storing their products, which can be toxic, flammable

IntelITSIS'2021: 2nd International Workshop on Intelligent Information Technologies and Systems of Information Security, March 24–26, 2021, Khmelnytskyi, Ukraine
EMAIL: drozd@ukr.net (O. Drozd); en.ivanova.ua@gmail.com (O. Ivanova); const-z@te.net.ua (K. Zashcholkin); romankev@scs.kpi.ua (V. Romankevich); yuliia.drozd@opu.ua (J. Drozd)
ORCID: 0000-0003-2191-6758 (O. Drozd); 0000-0002-4743-6931 (O. Ivanova); 0000-0003-0427-9005 (K. Zashcholkin); 0000-0003-4696-5935 (V. Romankevich); 0000-0001-5880-7526 (J. Drozd)

and explosive, add to the list of objects with increased risk, but far from exhausting it. Such a development of high-risk objects is an objective process already because mankind is not going to abandon them, despite the progressing statistics of man-made accidents and disasters.

The main feature of the described development is the growth of losses from accidents. Such growth is also characteristic of potential losses, i.e., losses from accidents that have not yet occurred, but will occur, according to statistics. The cost of loss is one of the factors in determining risk. Another factor is determined by the probability of an accident [1, 2]. Therefore, the problem of containing risks requires reducing this factor and is solved through the development of computer systems and information technologies implemented in them for this purpose. Such specialization of computer systems transforms them into safety-related systems, aimed at ensuring functional safety in the complex of both an object of increased risk and the system itself for preventing accidents and reducing losses in case of their occurrence [3, 4].

The main challenges in ensuring functional safety are associated with the occurrence of failures. Failure mitigation is based on the development of fault-tolerant solutions that ensure the continued correct functioning of the system under conditions of failure. Fault tolerance of circuit solutions is provided by using various types of redundancy and reconfiguration [5, 6].

The most common fault-tolerant solution is the majority structure, which in its simplest form contains three channels for solving the same task and a majority element that chooses the result by voting. In case of system failure in one channel, the result is determined by the sign of the coincidence of the results of the other two channels [7].

Is it enough to build a system that resists one failure? Typically, a single channel failure is much more likely than a dual channel failure. In safety-related systems, this rule is challenged, taking into account common cause failures that can occur when copying incorrect solutions, for example, design errors [8, 9].

In the case when the channels of the majority system use the same software containing an error, the failure can appear the same in all channels and determine the wrong result. In this regard, international standards regulating the provision of functional safety impose restrictions on copying solutions [10, 11] and propose to use multi-version technologies with an extended set of types of diversity to eliminate common causes [12, 13].

However, copying solutions is not the only source of multiple failures. Another equally dangerous source is associated with the problem of hidden faults. This problem is inherent only in critical applications, in which the operating mode is divided into normal and emergency. The problem is the accumulation of faults during a long normal operation in the absence of input data, which can manifest these faults in the form of error results. Accumulated faults can appear on the emergency input data. The number of faults manifested can exceed the capabilities of the fault-tolerant circuit in countering them and lead to a violation of the functional safety of both the system and the control object. It should be noted that hidden faults do not create problems for ordinary computers that operate in one mode and retain the latent nature of the malfunction throughout the entire mode [14, 15].

Development of components for modern systems of critical application widely uses FPGA design (Field Programmable Gate Array), which combines the advantages of a hardware solution and programmable logic [16, 17].

The programmability of digital circuits carries the additional risk of multiple failures associated with breaches of information security, including cyberattacks carried out by botnets. In this area, functional safety, in particular, the integrity of critical systems, must be ensured by methods and means of information security [18, 19].

A feature of FPGA projects is their LUT-oriented architecture, which inherits the problem of hidden faults in using the memory of LUT (Look-Up Table) units [20, 21].

The problem of hidden faults has not received a response in international standards for ensuring functional safety and is better known for unsuccessful attempts to solve it using simulation modes. These modes are aimed at detecting hidden faults by recreating emergency conditions and have repeatedly created a real threat to the functional safety of critical systems and facilities. Emergency conditions occurred as a result of unauthorized activation of simulation modes through the fault of the operator or due to a malfunction [22].

The planned use of simulation modes is also associated with a certain danger, since it involves the shutdown of emergency protection, which was one of the causes of the Chernobyl disaster [23].

The use of dangerous simulation modes indicates the importance of the problem of hidden faults, and also indicates the distrust that manifests itself in relation to fault-tolerant solutions. This distrust is based on the fact that a fault-tolerant solution does not become fail-safe when there is a deficiency of checkability, which is important to counter the accumulation of faults. Ignoring checkability can create a misconception about the sufficiency of information for assessing the fail-safety of a fault-tolerant circuit design [24, 25].

The fault tolerance of the circuit is laid in the design process in relation to a limited number of faults. This fact conflicts with a less limited set of faults that can be accumulated in circuits of critical application over long-term normal mode.

In operating mode, the checkability of circuits does not attract attention, since in ordinary computers it leads only to a slight increase in errors, the cost of which is usually low. In critical systems, checkability of circuits becomes an important condition for ensuring functional safety [26].

This paper is aimed at assessing the checkability of digital circuits in that part of it, which is important for ensuring fail safety in fault-tolerant FPGA components of safety-related systems.

Section 2 identifies the issue of assessing the checkability of digital circuits in FPGA components of safety-related systems. Section 3 describes a method for assessing checkability important for fail-safe FPGA components in critical applications. Section 4 presents the results of experiments on assessing the checkability of digital circuits using the example of a 4-bit multiplier implemented in an FPGA project for operation in normal and emergency modes.

2. Definition of problem

Checkability of digital circuits is best known in its simplest form, i.e. testability, which evaluates the possibility of developing tests for detecting circuit faults based on the analysis of its controllability and observability [27, 28].

As a rule, test sequences for information inputs of a circuit are determined based on the set of all possible input data. Therefore, the testability of a circuit is completely determined by its structure, i.e. is structural checkability. Testability can also be classified as logical checkability in view of its orientation toward logical checking aimed at error detection in binary codes.

In the operating mode, the logic checkability of a digital circuit evaluates its ability to exhibit a fault in the form of a result error on the input data used. For this reason, circuit checkability adds dependency on the input. The result error, the appearance of which is ensured by the checkability of the digital circuit, is a prerequisite for detecting a fault when performing on-line testing of the circuit using logical checking methods [29]. On-line testing can also be carried out by monitoring energy parameters. However, such monitoring can only detect failures that cause significant changes in temperature or current consumption that go beyond the operating values of these parameters under normal conditions [30, 31]. It should be noted that on-line testing, directed in conventional computers to control the trustworthiness of calculated results, performs this function as the main one only in emergency mode. The normal mode of critical systems is used primarily for clearing circuits from faults. In this case, on-line testing performs the function of testing, but under limited conditions of using operating input data as test sequences [32, 33].

Safety-related systems make the checkability of digital circuits dependent on various inputs used in normal and emergency modes. Different input data determine different checkability of the system in these modes and thus create conditions for the accumulation of hidden faults that cause a problem in ensuring the fault tolerance of system components with the onset of an emergency mode [34, 35].

A digital circuit implemented in an FPGA project with a LUT-oriented architecture represents computations in the form of decomposition with their division into logical functions of several variables. These functions are implemented by LUT units. The description of logical functions is written into the memory of the LUT units in the form of a program code when programming an FPGA microcircuit. The program codes of the LUT units form the program code of the FPGA project. Variables at the input of the LUT unit constitute the address at which the corresponding value of the function is read to the output of the LUT unit from its memory. The most commonly used LUT-oriented architecture is based on LUT units, which contain 4 inputs: A, B, C, D and 16 memory bits [36, 37].

The checkability of FPGA components must be considered taking into account possible faults in a digital circuit with a LUT-oriented architecture.

Possible faults of the LUT unit are determined by its circuit, which contains a register (LUT memory) for storing the program code and a multiplexer that selects a bit of the program code from the LUT memory at a given address. The multiplexer consists of switches that perform the function of selecting one bit from two directions to one. The switches are controlled using the inputs of the LUT unit. The program codes of the LUT units form the program code of the FPGA project, which is verified using a checksum. For this reason, the memory of the LUT units is checkable and does not pose a threat to the fault tolerance of digital circuits and the functional safety of critical systems.

However, checking the program code does not indicate faults in the multiplexer switches. These faults can distort the values of the read bits of the LUT memory and the addressing to them. Constant faults at the information inputs of the switches can distort the values of the read bits, and faults in the control inputs lead to addressing errors. Distortion of the address of the read bit causes an error when the values of the bits located at the correct and corrupted address do not match. All described faults have external manifestations that can be identified with faulty bits of the LUT memory. Therefore, the checkability of a digital circuit with a LUT-oriented architecture is further discussed in relation to the memory LUT bits.

In a logical aspect, the checkability of a digital circuit is determined taking into account the manifestation of its faults in the form of errors in the calculated result. Faults that do not cause errors and circuits containing these faults are not checkable. However, the question about the lack of checking for such faults usually does not arise, as well as questions about the influence of these faults on the calculation results. As a rule, these faults are excluded from the field of view as they do not have any effect on the computation process.

The use of FPGA components in critical systems that differ in the presence of two modes of operation significantly changes the attitude towards faults, since the checkability of the circuit plays a different role in normal and emergency modes: positive and negative, respectively. Indeed, improving the checkability of a circuit is the best manifestation of its faults. Using the normal mode to clear the circuit from faults motivates an increase in its checkability. The best conditions for ensuring functional safety are the absence of failures, or at least the manifestation of malfunctions in the form of errors in emergency mode. However, the best concealment of faults is achieved under conditions of poor digital circuit checkability. High checkability of the circuit contributes to the best manifestation of faults in the form of errors. First of all, this drawback relates to transient faults that occur much more often than failures but, like failures, cause errors that reduce the trustworthiness of the calculated results [38].

In safety-related systems, the contradiction between the checkability of the circuit and the trustworthiness of the results is no longer mutually exclusive due to the different input data used in normal and emergency modes. Different checkability of these modes creates conditions for masking a part of accumulated faults with the onset of an emergency mode. This feature reduces the requirements for fault tolerance of circuits from the standpoint of ensuring their fail-safety. However, such masking does not provide a guarantee of limiting the number of accumulated faults that could be sufficient for a fully fault-tolerant and therefore safe operation of a safety-related system.

Therefore, the issue of the efficiency in the checkability of the digital circuit in relation to its fail-safety remains open, taking into account the fault tolerance, which is provided to counter a limited number of failures, and in connection with the possibility of accumulating faults. The checkability assessment of a digital circuit should be structured to highlight the portion of it that is important to the fail-safety of a fault-tolerant solution.

3. Assessment of checkability of circuits with LUT-oriented architecture in critical applications

The checkability of a digital circuit with a LUT-oriented architecture is assessed by the ratio of the number of checkable bits of the LUT memory to their total number. The main property of the checkable bits of the LUT memory is the openness to demonstrate their correct or faulty state.

The proposed method evaluates the checkability of a digital circuit with LUT-oriented architecture in the part that supports the fail-safety of fault-tolerant FPGA components in safety-related systems.

The analysis of the checkability of a circuit from the standpoint of its fail-safety requires attributing to the checkable bits the part of the LUT memory which directly counteracts the accumulation of faults that manifest themselves in emergency mode reducing the fault tolerance of the circuit.

The ability of the checkable bits of the LUT memory to demonstrate correct circuit operation or to malfunction in the form of a result error shall be used in a normal mode. This requirement assigns the checkable bits to the set that is addressed in this mode. However, the primary value of checkable bits lies in their ability to ensure that the circuit functions correctly in an emergency. Therefore, checkable bits that provide fail-safety of fault-tolerant FPGA components should be looked for in the part of the LUT memory that is addressed in both modes of the safety-related system.

At the same time, addressing the memory LUT bit does not mean using this bit to form the result obtained at the output of the circuit. An indication of the use of a memory LUT bit is the influence of its distortion caused by a fault on the correctness of the result.

The checkability of a digital circuit and its elements should be assessed in terms of controllability and observability.

The controllability of the LUT memory is provided in its addressable bits. The observability of the LUT memory bit is manifested in its use for generating a result when bit corruption determines an erroneous result. Since the use of the LUT memory bit for the formation of the result implies addressing this bit, the observed bit is also controllable and, therefore, checkable.

The fail-safety of fault-tolerant circuit is provided in the memory LUT bits which are observable in both normal and emergency mode. The observability of the memory LUT bits in normal mode counteracts the accumulation of hidden faults. The observability of the LUT memory bits in emergency mode classifies them among the many important for ensuring the functional safety of the FPGA component.

Such checkable memory LUT bits are important for the functional safety of the critical system, since they do not pose the problem of hidden faults. At the same time, ensuring functional safety requires first of all pay attention to the memory LUT bits in which this problem can manifest itself. These problem bits are observable only in emergency mode, which is manifested in their influence on the generated computation result. In normal mode, they are not used to generate the result and are not addressed.

The proposed method defines two sets, O_{NE} and O_{EN}, of memory LUT bits, which are important for fail-safety of fault-tolerant FPGA components. These sets consist of bits that support and, on the contrary, violate fail-safety of fault-tolerant components, respectively.

The method simulates the operation of an FPGA component in normal and emergency modes. The initial data for simulating are the description of the circuit and the input data, which are fed to the inputs of the circuit in normal and emergency modes.

To determine the O_{NE} and O_{EN} sets, it is enough to evaluate the observability of all memory LUT bits in each operating modes of the FPGA component. Bit controllability is not directly considered in defining these sets. However, estimating the controllability of the memory LUT bits reduces the amount of computation, since it reduces the number of bits for which a more laborious procedure for determining their observability is performed.

The controllability of all bits of the LUT memory is determined by simulating the computations performed by the FPGA component on all normal and emergency mode inputs. For each LUT unit, all addressable memory bits are marked.

The observability of each bit of the LUT memory is determined independently of the other bits by simulating the operation of the FPGA component with a correct and incorrect value of this bit on all inputs. For each LUT unit, all memory bits that take part in the formation of the computation result are marked.

The method evaluates the controllability of the memory LUT bits in each of the modes and generates data arrays marking C_N and C_E of the set of bits controlled in normal and emergency mode, as well as the set of C_{NE} and C_{EN} bits controlled in both modes and only in emergency mode, respectively.

The resulting sets of controlled bits of the LUT memory are in the following dependencies: $C_{NE} \cap C_{EN} = \varnothing$, $C_{NE} \cup C_{EN} = C_E$.

Observability is evaluated for all controllable bits of the set C_E, taking into account whether these bits belong to the set C_{NE} or C_{EN}. The simulation results define data arrays marking the O_E set of all memory LUT bits observed in emergency mode, as well as the $O_{N\text{-}E}$ and $O_{E\text{-}E}$ bit sets observed on normal and emergency mode inputs, respectively. The O_E set combines all of the memory LUT bits important to the safety of the FPGA component.

The sets $O_{N\text{-}E}$ and $O_{E\text{-}E}$ are used to define the memory LUT bits that provide checkability and make it deficient in the fail-safety of the fault-tolerant circuit: $O_{NE} = O_{N\text{-}E} \cap O_{E\text{-}E}$, $O_{EN} = O_{E\text{-}E} \setminus O_{N\text{-}E}$.

The resulting sets of memory LUT bits show the following relationships: $O_{NE} \cap O_{EN} = \varnothing$, $O_{NE} \cup O_{EN} = O_E$.

The O_E set combines all of the memory LUT bits that are important to the fail-safety of the fault-tolerant FPGA components. Therefore, the proposed method evaluates the checkability of digital circuits and its deficiency in relation to functional safety with respect to the O_E set.

Checkability of the FPGA component determines the degree in fail-safety of the fault-tolerant circuit using the following formula: $A = |O_{NE}| / |O_E|$.

The checkability deficit of a FPGA component defines the degree of violation in fail-safety of the fault-tolerant circuit as follows: $B = |O_{EN}| / |O_E|$.

The method forms both indicators A and B, complementing each other to one: $A + B = 1$.

4. Program implementation of the method

The proposed method is considered on the basis of its program implementation, developed using the Delphi 10 Seattle demo version [39].

The developed program makes it possible to assess the checkability of a circuit with a LUT-oriented architecture in terms of ensuring its fail-safety. As initial data, the program uses the description of the circuit and the ranges of the input data arriving at the inputs of the circuit. The description of the circuit contains a list of LUT units with an indication of the program codes, as well as the inputs of the circuit and outputs of the LUT units that are connected to the inputs of other LUT units. In addition, the circuit inputs and LUT units connected to its outputs are described.

The input ranges used in normal and emergency modes are separated by a threshold that places a limit on the computed results. A result that is less than the threshold puts the input data in normal mode. Reaching and exceeding the threshold marks the beginning of the emergency mode.

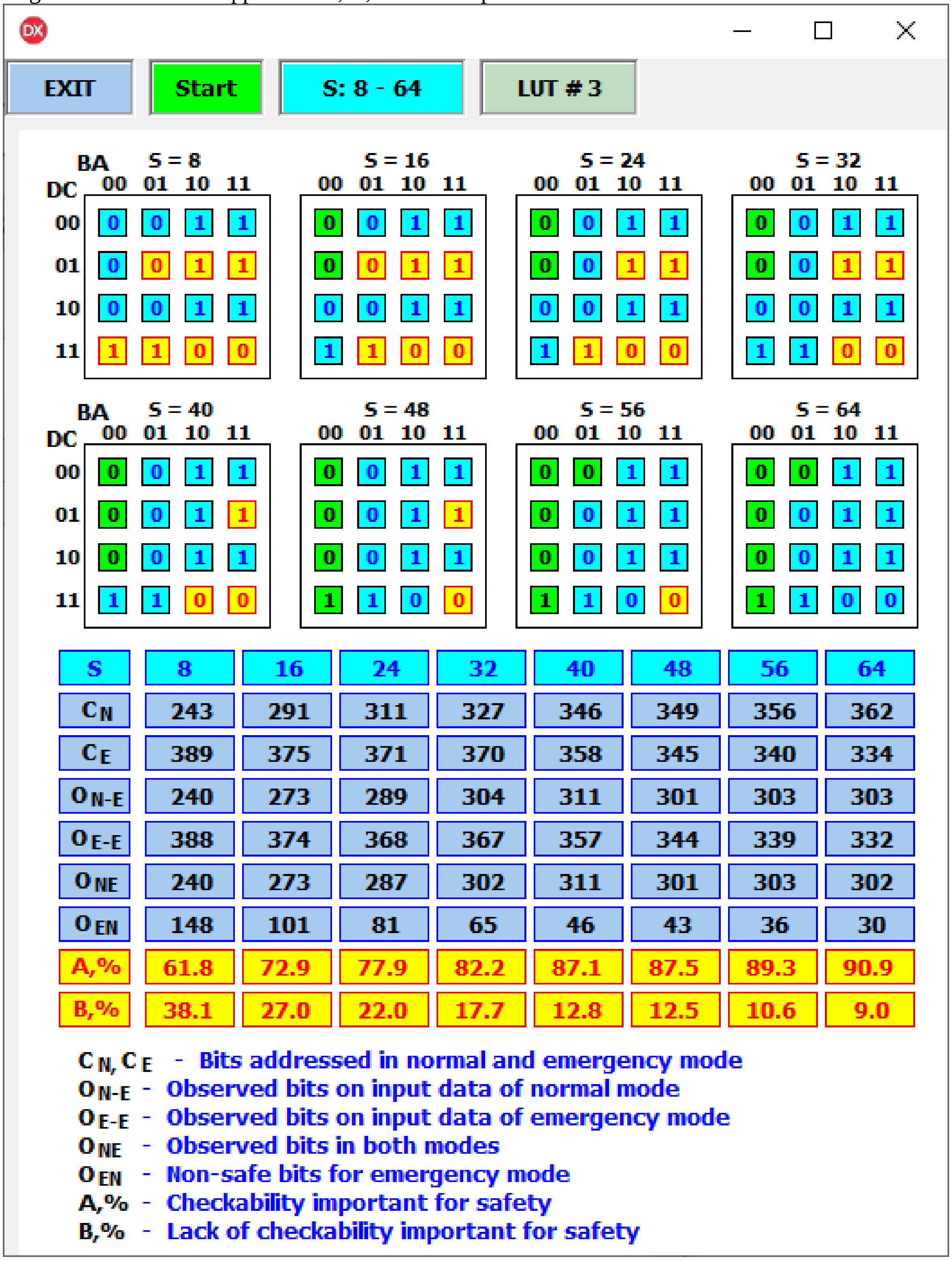
The program implementation of the method was tested on FPGA circuits of library arithmetic devices. As an example, checkability score is shown for a 4-bit iterative array multiplier, the circuit of which is implemented in the Intel Cyclone 10 LP FPGA chip: 10CL025YU256I7G using CAD Quartus Prime 20.1 Lite Edition [40, 41].

The resulting circuit performs multiplication of binary codes and uses 30 LUT units for this. The program allows to compare simulation results for eight different threshold values. As the initial data, the base value of the threshold is set, according to which seven more smaller thresholds are determined at equal intervals. For each value S of the threshold, the normal and emergency input data are determined by comparing the calculated result with this threshold, and an independent simulation of the circuit is performed. The normal mode is determined starting from the zero-product value.

Fig. 1 shows the main panel of the program for the basic value of the threshold S = 64, which defines 8 values with the same interval: 8, 16, 24, …, 64. The panel contains the control keys “EXIT” and “START” to exit the program and start simulation, respectively. Next, the lowest and highest threshold value is indicated in the form “S: 8 - 64”. The next key “LUT # 3” is used to sequentially view the memory of all LUT units in the circuit. In this case, the key defines LUT unit with number 3 for viewing its memory. When this key is pressed, the number of the viewed LUT unit is increased by one. After the last number, the key takes on the initial value “LUT # 1”.

The memory of the selected LUT unit is shown for all eight threshold values as bit matrices. The memory LUT matrices are arranged in two rows and labeled with a threshold value from S = 8 to

S = 64. The memory bits are shown in the matrix as squares containing their values, and are numbered using the address code supplied to A, B, C and D inputs of the LUT unit.



S = 8

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

S = 16

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

S = 24

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

S = 32

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

S = 40

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

S = 48

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

S = 56

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

S = 64

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

S	8	16	24	32	40	48	56	64
C_N	243	291	311	327	346	349	356	362
C_E	389	375	371	370	358	345	340	334
O_{N-E}	240	273	289	304	311	301	303	303
O_{E-E}	388	374	368	367	357	344	339	332
O_{NE}	240	273	287	302	311	301	303	302
O_{EN}	148	101	81	65	46	43	36	30
A,%	61.8	72.9	77.9	82.2	87.1	87.5	89.3	90.9
B,%	38.1	27.0	22.0	17.7	12.8	12.5	10.6	9.0

C_N, C_E - Bits addressed in normal and emergency mode
O_{N-E} - Observed bits on input data of normal mode
O_{E-E} - Observed bits on input data of emergency mode
O_{NE} - Observed bits in both modes
O_{EN} - Non-safe bits for emergency mode
A,% - Checkability important for safety
B,% - Lack of checkability important for safety

Figure 1: Main panel of the method program implementation

The rows and columns of the matrix are numbered with binary values 00, 01, 10, 11 of the high and low half of the address at inputs D, C and B, A, respectively. For example, the upper left bit of the LUT memory located at the intersection of row 00 and column 00 is $0000_2 = 0$, and the lower right bit number is determined by the intersection of row 11 and column 11 as $1111_2 = 15$. For all threshold values, the matrices show the same values of the memory bits of the selected LUT unit, but can be colored differently depending on their use in normal and emergency mode. The squares for bits addressable only in normal and only in emergency mode are colored green and yellow, respectively. The “Aqua” color bits are addressed in both modes. The values of the memory LUT bits that are observable in both modes and only in emergency mode are highlighted in blue and red, respectively. All of these highlighted bits are important to the fail-safety of the circuit.

Bits with values highlighted in blue are checkable, while bits highlighted in red violate fail-safety. They are dangerous because they can accumulate hidden faults that reduce the fault tolerance of the FPGA component.

With an increase in the threshold S, the number of checkable bits increases and the number of dangerous bits, causing a deficit of checkability, decreases.

In the case of S = 8, the LUT memory contains 9 checkable bits: 0 – 4, 8 – 11 and 7 dangerous bits: 0 – 4, 8 – 11. All 16 bits are important for the safety of the circuit.

The threshold S = 16 reduces the number of important bits, excluding from their set bits 0 and 4, which are addressed only in normal mode. In addition, the previously unsafe bit 12 becomes checkable. Raising the threshold to values S = 24 and S = 32 continues the conversion of dangerous bits into checkable bits for numbers 5 and 13.

Each of the thresholds S = 40, S = 48, and S = 56 reduces the set of important bits, excluding bits 8, 12, and 1. The reduction in the number of important bits occurs by reducing the number of dangerous bits by converting bits 6, 14 and 7 into checkable. In case of threshold S = 64, LUT memory is completely cleared of dangerous bits. All important bits become checkable.

The panel below shows the main simulation results, which contain the cardinalities for the sets C_N, C_E, $O_{N\text{-}E}$, $O_{E\text{-}E}$, O_{NE}, O_{EN}, as well as the A and B scores of checkability and its deficiency important to safety (expressed as a percentage).

As the threshold S increases from 8 to 64, the number of LUT memory bits addressed in normal and emergency modes increases and decreases from 243 to 362 and from 389 to 334, respectively. The number of LUT memory bits observed on the normal and emergency input data increases and decreases in the range 240 – 303 and 388 – 332, respectively. The number of checkable and dangerous bits of the LUT memory increases and decreases in the range 240 – 302 and 148 – 30, respectively. Checkability increases from 61.8% to 90.9%, and its deficit decreases from 38.1% to 9.0%.

The performed modeling shows the limitation of the checkability of FPGA components in ensuring the fail-safety of fault-tolerant circuits and the possibility of reducing its deficit with an increase in the range of input data used in the normal mode.

5. Conclusions

Safety-related systems ensure their own functional safety and the safety of controlled objects using fault-tolerant solutions, for which multiple failures are the most dangerous. One of the sources of multiple failures is associated with the problem of hidden faults, which can be accumulated during a prolonged normal mode and reduce the fault tolerance of circuits with the onset of emergency conditions. The accumulation of hidden faults in the normal mode and their manifestation in the most critical emergency mode occurs due to the limited checkability of the circuits on the input data of the normal mode and the difference in checkability demonstrated by the circuits in the normal and emergency modes because of different input data. Thus, the checkability of circuits is an important argument in ensuring the functional safety of critical systems and a prerequisite for transforming fault-tolerant solutions into fail-safe ones.

FPGA component design, which is widely used for critical applications, inherits the problem of hidden faults in LUT units of digital circuits with LUT-oriented architecture. The programmable memory of the LUT units is checked with a checksum within the entire FPGA project. However, the reading of the bits of this memory can be corrupted by faults that do not affect the checksum and are hidden in normal mode.

The proposed method evaluates the checkability of the circuit in these memory bits in terms of ensuring the fail-safety of fault-tolerant FPGA components. The method defines the set of memory LUT bits that are important to fail-safety of fault-tolerant circuits. These bits are split into two sets, which provide and, on the contrary, violate the fail-safety of the circuit. Bits of the first set are observable in both normal and emergency modes. The ratio of their number to the total number of important bits determines the checkability of the circuit, which prevents the accumulation of hidden faults that appear in emergency mode, and thus ensures the fail-safety of FPGA components. The second set contains memory LUT bits that are observable only in emergency mode and constitute a deficiency of checkability required to transform a fault-tolerant circuit into a fail-safe circuit.

6. References

[1] T. Aven, E. Zio, Foundational issues in risk analysis, Risk Analysis, 34 (7) (2014) 1164-1172.

[2] A. O'Connor, A. Mosleh, A general cause based methodology for analysis of common cause and dependent failures in system risk and reliability assessments, Reliability Engineering & System Safety 145 (2016) 341–350.

[3] J. Mayaka, J.C. Jung Complexity reduction of the Engineered Safety Features Component Control System, Nuclear Engineering and Design, 331 (2018) 194–203.

[4] Safety Classification for I&C Systems in Nuclear Power Plants – Current Status & Difficulties. Report No. 2015/008. Produced by: World Nuclear Association, 2015.

[5] S.F. Tyurin, Investigation of a Hybrid Redundancy in the Fault-Tolerant Systems, Radio Electronics, Computer Science, Control 2 (2019) 23–33. doi:10.15588/1607-3274-2019-2-3.

[6] I. Atamanyuk, Y. Kondratenko, Computer's analysis method and reliability assessment of fault-tolerance operation of information systems, in: CEUR Workshop Proceedings, vol. 1356, 2015, pp. 507–522.

[7] S. Tyurin, A Quad CMOS gates checking method, International Journal of Computing 18(3) (2019) 258–264.

[8] M. Rausand, Risk Assessment. Common Cause Failures. NTNU, Trondheim.

[9] E. Brezhnev, An approach for assessing risk of common cause failures in critical infrastructure, Information & Security, 28(1) (2012) 199–210.

[10] M. Kumar, A. Kabra, G. Karmakar, P.P. Marathe, A Review of Defences against Common Cause Failures, in: Reactor Protection Systems, 4th International Conference on Reliability, Infocom Technologies and Optimization, Noida, India, 2015, pp. 1–5.

[11] J. R. Belland, Modeling common cause failures in diverse components with fault tree applications, in: Annual Reliability and Maintainability Symposium, Orlando, FL, USA, 2017.

[12] M. Yastrebenetsky, V. Kharchenko, (Ed.), Nuclear Power Plant Instrumentation and Control Systems for Safety and Security, book series AEEGT, IGI Global, 2014. doi:10.4018/978-1-4666-5133-3.

[13] H. Asad, I. Gashi, Diversity in Open Source Intrusion Detection Systems. In: Computer Safety, Reliability, and Security, LNCS, 11093, Springer, 2018, pp. 267–281.

[14] O. Drozd, V. Antoniuk, V. Nikul, M. Drozd, Hidden faults in FPGA-built digital components of safety-related systems, in: Proceedings of the International Conference TCSET, Lviv-Slavsko, Ukraine, 2018, pp. 805–809. doi:10.1109/TCSET.2018.8336320

[15] O. Drozd, M. Kuznietsov, O. Martynyuk, M. Drozd, A method of the hidden faults elimination in FPGA projects for the critical applications, in: Proceedings of the International Conference DESSERT, Kyiv, Ukraine, 2018, pp. 231–234. doi:10.1109/DESSERT.2018.8409131

[16] I. Ahmed, J. Jung, G. Heo, Design verification enhancement of field programmable gate array-based safety-critical I&C system of nuclear power plant, Nuclear Engineering and Design 317 (2017) 232–241.

[17] Application of Field Programmable Gate Arrays in Instrumentation and Control Systems of Nuclear Power Plants, IAEA NUCLEAR ENERGY SERIES No. NP-T-3.17, 2016.

[18] S. Lysenko, K. Bobrovnikova, S. Matiukh et. al., Detection of the botnets' low-rate DDoS attacks based on self-similarity, International Journal of Electrical and Computer Engineering 10(4) (2020) 3651–3659.

[19] S. Lysenko, K. Bobrovnikova, O. Savenko, A Botnet Detection Approach Based on the Clonal Selection Algorithm, in: Proceedings of the 9th IEEE International Conference DESSERT, Kyiv, Ukraine, 2018, pp. 424–428. doi: 10.1109/DESSERT.2018.8409171.

[20] Z. T. Sworna, M. U. Haque, H. M. H. Babu et. al., An Efficient Design of an FPGA-Based Multiplier Using LUT Merging Theorem, in: Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Bochum, Germany, 2017, pp. 116–121.
[21] O. Drozd, V. Romankevich, M. Kuznietsov et. al., Using natural version redundancy of FPGA projects in area of critical applications, in: Proceedings of the International Conference DESSERT, 2020, pp. 58–63. doi: 10.1109/DESSERT50317.2020.9125050.
[22] D. Gillis, The Apocalypses that Might Have Been, 2007. URL: https://www.damninteresting.com/the-apocalypses-that-might-have-been/
[23] The true toll of the Chernobyl disaster, 2019. URL: https://www.bbc.com/future/article /20190725-will-we-ever-knowchernobyls-true-death-toll.
[24] T. Hovorushchenko, Methodology of Evaluating the Sufficiency of Information for Software Quality Assessment According to ISO 25010, Journal of Information and Organizational Sciences 42(1) (2018) 63–85 doi: 10.31341/jios.42.1.4.
[25] T. Hovorushchenko, A. Herts, Ye. Hnatchuk. Concept of Intelligent Decision Support System in the Legal Regulation of the Surrogate Motherhood. CEUR-WS, 2019, vol. 2488, pp. 57–68.
[26] O. Drozd, K. Zashcholkin, R. Shaporin, J. Drozd, Y. Sulima, Development of ICT Models in Area of Safety Education, in: Proceedings of the EWDTS, 2020, pp. 115–119. doi: 10.1109/EWDTS50664.2020.9224861.
[27] T. Shah, A. Matrosova, M. Fujita et. al., Multiple Stuck-at Fault Testability Analysis of ROBDD Based Combinational Circuit Design, Journal of Electronic Testing, 34 (2018) 53–65.
[28] T. Chowdary, M. Prasad, A Short Paper on Testability of a SoC, International Journal of Engineering & Technology 7(3.12) (2018) 326.
[29] O. Drozd, I. Perebeinos, O. Martynyuk et. al., Hidden fault analysis of FPGA projects for critical applications, in: Proceedings of the IEEE International Conference TCSET, 142, 2020. doi: 10.1109/TCSET49122.2020.235591.
[30] G. Farkas, Z. Sarcany, M. Rencz, Structural Analysis of Power Devices and Assemblies by Thermal Transient Measurements. Energies 12 (2019) 2696.
[31] G. Farkas, D. Schweitzer, Z. Sarkany, M. Rencz, On the Reproducibility of Thermal Measurements and of Related Thermal Metrics in Static and Transient Tests of Power Devices, Energies 13 (2020) 557.
[32] A. Drozd, J. Drozd, S. Antoshchuk et. al., Objects and Methods of On-Line Testing: Main Requirements and Perspectives of Development, in: Proceedings of the EWDTS, 2016, pp. 72–76. doi:10.1109/EWDTS.2016.7807750.
[33] D. Coppad, D. Sokolov, A. Bystrov, A. Yakovlev, Online Testing by Protocol Decomposition, in: Proceedings of the 12th IEEE International Symposium IOLTS, Como, Italy, 2006, pp. 263–268. doi:10.1109/IOLTS.2006.45.
[34] J. Drozd, A. Drozd, M. Al-dhabi, A resource approach to on-line testing of computing circuits, in: Proceedings of the EWDTS, 2015, pp. 276–281. doi:10.1109/EWDTS.2015.7493122.
[35] O. Drozd, K. Zashcholkin, O. Martynyuk, O. Ivanova, J. Drozd. Development of Checkability in FPGA Components of Safety-Related Systems. CEUR Workshop Proceedings, vol. 2762, pp. 30-42 (2020). URL: http://ceur-ws.org/Vol-2762/paper1.pdf.
[36] M. Ebrahimi, R. Sadeghi, Z. Navabi, LUT Input Reordering to Reduce Aging Impact on FPGA LUTs, in: IEEE Transactions on Computers 69(10) (2020) 1500–1506.
[37] Intel FPGA Architecture, 2019. URL: https://www.intel.com/content/dam/www/programmable /us/en/pdfs/literature/wp/wp-01003.pdf.
[38] N. Arya, A. P. Singh, Fault Tolerant System for Embedded System Architecture, International Journal of Engineering and Technology (IJET) 9(3S) 2017 93–97.
[39] Delphi 10 Seattle: Embarcadero https://www.embarcadero.com/docs/datasheet.pdf.
[40] Intel Cyclone 10 LP Core Fabric and General Purpose I/Os Handbook, 2020. URL: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/cyclone-10/c10lp-51003.pdf.
[41] Intel Quartus Prime Standard Edition User Guide, 2020. URL: https://www.intel.com/content/www/us/en/programmable/documentation/grc1529967026944.html