Augmented Checkability of LUT-oriented Circuits in FPGA-based Components of Safety-Related Systems

Oleksandr Drozd[a], Illya Baskov[a], Oleksandr Martynyuk[a], Kostiantyn Zashcholkin[a] and Myroslav Drozd[a]

[a] *Odesa Polytechnic National University, Ave. Shevchenko 1, Odesa, 65044, Ukraine*

Abstract

The paper focuses on extending the checkability of LUT-oriented circuits for FPGA-based (Field Programmable Gate Array) digital components. The main provisions of the article concentrate on the task of ensuring the checkability for critical computer systems. As critical systems, the article considers computer systems that control high-risk facilities or ensure the functional safety of such facilities. Traditionally, approaches that are based on fault tolerance have been used to ensure the functional safety of computer systems. The specific of critical systems is that these systems must potentially operate in two modes: normal and emergency. Critical systems operate in normal mode for most of their life cycle. This mode is a long time and stable in the nature of the data processed. In this case, the input data of the system have low change activity. This leads to a decrease in the checkability of the circuits and, consequently, to the accumulation of hidden faults. When an emergency mode occurs, the activity of the input data of the system increases. The checkability of the circuit also increases. But, on the other hand, it leads to the fact that hidden faults appear. This in turn reduces the fault tolerance of the system. The paper proposes a method of checkability assessment, which analyzes faults in a circuit built on the basis of elementary LUT units. The method identifies circuit faults by faults of memory bits of LUT units. The method analyzes memory bits that are essential to the operation of the critical system in emergency mode. The complete set of memory bits is divided into two subsets: bits that are checkable in normal mode and, bits that are not checkable in this mode. The method extends the checkability of circuits by identifying the situation when the fault is caused by bits from both subsets (checkable and not checkable). Detecting this relationship of bits from different subsets gives the possibility to expand the checkability of the circuit. Experiments have been carried out to confirm the expansion of checkability for the implementation of the iterative array multiplier on FPGA.

Keywords

Safety-related system, FPGA-based component, LUT-oriented circuit, LUT memory, fault tolerance, hidden faults, fail-safety, augmented checkability

1. Introduction

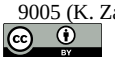
The observed priority development of critical infrastructures in energy, chemical production, transport and other important industries is due to their role in ensuring energy, food and other types of security, which create the basis for the successful evolution of mankind.

The natural processes are an increase in the number and capacity of power plants and power grids, toxic and fire hazardous industries, and high-speed transport. These technical facilities belong to the class of high-risk facilities. Operation of such facilities requires ensuring the safety of critical

IntelITSIS'2022: 3nd International Workshop on Intelligent Information Technologies and Systems of Information Security, March 23–25, 2022, Khmelnytskyi, Ukraine

EMAIL: drozd@ukr.net (O. Drozd); illyabaskov@gmail.com (I. Baskov); anmartynyuk@ukr.net (O. Martynyuk); const-z@te.net.ua (K. Zashcholkin); myroslav.drozd@opu.ua (M. Drozd)

ORCID: 0000-0003-2191-6758 (O. Drozd); 0000-0002-3517-6773 (I. Baskov); 0000-0003-1461-2000 (O. Martynyuk); 0000-0003-0427-9005 (K. Zashcholkin); 0000-0003-0770-6295 (M. Drozd)



CEUR Workshop Proceedings (CEUR-WS.org)

infrastructure. As well as to increase functional safety of computer systems that control high-risk objects. Such systems are commonly referred to as safety-related systems.

Computer systems of this kind are needed to control and monitor high-risk facilities, as well as to prevent accidents [1]. This specialization of computer systems aims to contain risks, which are characterized firstly by the probability of an accident, and secondly by the cost of their consequences [2].

Improvements in critical infrastructures continually increase these risks in the cost of accident consequences. This leads to increasing requirements that are placed on the functional safety of critical systems. The final goal of which is to reduce the probability of accidents. In accordance with the specialization, safety-related systems are endowed with a significant difference, which consists in their design for operation in two modes: normal and emergency [3].

The main threat to functional safety comes from failures, which determine the need to build safety-related systems based on fault-tolerant solutions [4].

However, fault-tolerant structures provide operation under a given amount of failures. Usually this amount is one or two failures. But under multiples of failures, such systems cannot operate properly. Practice shows that often multiple failures are associated with hidden faults. This is a consequence of low checkability in normal mode of operation of the system [5].

Current approaches to computer system design do not take into account the possibility of hidden faults.

Low checkability in the conditions of hidden faults leads to multiple failures. FPGA (Field Programmable Gate Array) chips are widely used as an element base for modern computer systems. Chips of this type demonstrate the problem of hidden faults in the architecture based on the use of elementary LUT (Look-Up Table) units [6, 7].

The purpose of this paper is to study and evaluate the checkability of safety-related systems based on FPGA chips.

Further, the paper is organized as follows.

Section 2 analyzes existing work that confirms the relevance of the problem of checkability for FPGA components that are part of safety-related systems.

In Section 3 we present a method for evaluating checkability for an FPGA circuit in which two characteristic modes of operation are possible: normal and emergency.

Section 4 describes the experimental evaluation of the proposed method. Here the checkability of the LUT-oriented matrix multiplier circuit is studied.

2. Related works

The threat to H-fault-tolerant structures, emanating from sources of K-multiple failures, $H < K$, is quite real, given that the parameter H, which is formed in the design of a fault-tolerant structure, as a rule, does not exceed two failures and does not increase throughout the entire life cycle. The limitation of the parameter H is usually taken taking into account the significant complication of the circuit design and an equally significant decrease in the probability of occurrence of independent failures with an increase in their multiplicity.

However, such a view is sinful of the substitution of concepts. The problem of multiple failures is not due to their occurrence, but to a massive manifestation that impedes the performance of the assigned task. In addition, multiple failures are usually accompanied by a common cause that calls into question their complete independence.

The issue of a common cause in the occurrence of failures is investigated in international standards that define the requirements for the functional safety of critical systems. Common cause failures are considered in relation to design errors, system defects in manufacturing, and the like, i.e., from the position of copying erroneous decisions [8]. Copying errors leads to multiple failures in duplicate circuits of fault-tolerant structures [9].

In order to eliminate failures of this kind, the standards require that the design of critical systems be limited to copying schematic solutions. This is achieved through multiversion approaches and the use of several types of diversity [10].

In the described case, failures both arise and simultaneously manifest themselves, which blurs the line between the occurrence and manifestation of failures and creates conditions for ignoring the difference between these processes.

Another manifestation of concept substitution is the attribution of common cause failures due to copying errors directly to functional safety issues, although they are primarily characteristic of duplicate circuits used in fault-tolerant structures.

From this point of view, hidden faults that can arise and accumulate at different points in time of a prolonged normal mode and manifest massively with the onset of an emergency mode show the problem of multiple failures from a new side, distinguishing between the occurrence and manifestation of failures.

In addition, hidden faults pose a problem only for safety-related systems and are completely harmless for computers operating in one operating mode, since they remain hidden throughout this entire mode.

Thus, the main dangerous process is not the occurrence of failures, but their manifestation, which is investigated and evaluated by the checkability of the circuit. This fault-indicating property of a circuit turns out to be as important to functional safety as fault tolerance, which fails to handle failures without considering checkability.

Testability is derived from testability, which depends completely on the structure of the digital device [11].

In the operating mode, this structural form also acquires a dependence on input data, which can contribute or, conversely, prevent the manifestation of faults. By the way a fault is manifested in the form of an error or a change in energy parameters, checkability can be used, respectively, in a logical or energy-oriented form [12].

Logical checkability, being the main one for digital circuits, regulates the possibilities of on-line testing in detecting errors caused by faults on base of methods and means of logical checking [13].

The development of critical infrastructures and their monitoring by safety-related systems led to the further evolution of checkability, which is associated with the separation of the operating mode into normal and emergency.

This separation leads to the separation of functions and parts of the circuit, as well as the input data, which are activated differently in these modes. All this is reflected in the checkability of digital circuits, which also becomes different in normal and emergency mode and, moreover, begins to play a different role in ensuring functional safety.

The ability of a circuit to fault manifestation defines its checkability as an important attribute for performing on-line testing functions and opposes the trustworthiness of calculated results, which decreases in the event of malfunctions in the form of errors.

In normal mode, checkability plays a positive role by ensuring that the circuit is cleared of faults by methods of on-line testing. With the onset of the emergency mode, checkability becomes a hindrance to functional safety, reducing the trustworthiness of the calculated results with errors caused by manifested faults.

Changing modes also affects on-line testing, which changes its purpose. The normal mode can be considered as a test mode, preceding the emergency mode as the main one from the point of view of the focus of critical systems on preventing accidents and reducing accident losses.

In this case, on-line testing should inherit the inherent purpose of testing, which is performed during the pauses of the main operations, i.e., to detect faults in digital circuits. However, this purpose is realized not on tests, but on operating inputs with low circuit checkability in normal mode and therefore creates a problem of hidden faults.

A long period in evolution of on-line testing associated with the development of totally self-checking circuits took place precisely for this purpose, although its real purpose is to check the trustworthiness of the calculated results [14].

Purpose substitution took place within the framework of exact data processing, which became the basis for the definition and construction of totally self-checking circuits that detect a fault at its first manifestation in the form of an error.

Typically, the first error is caused by a transient fault due to its more frequent occurrence than a permanent fault.

Transient fault detection is not important from the point of view of circuit performance, but is important in checking the trustworthiness of the result.

For exact data, both of the named purposes of on-line testing are indistinguishable, since the detected error is a symptom of both the detection of faults and the calculation of an unreliable result. Indeed, any error in an exact number containing only the most significant bits makes the result unreliable.

In the processing of approximate data, the purposes diverge in accordance with the structure of the approximate data, which contains the most significant and least significant bits in the high and low positions, respectively.

In these bits, faults occur errors, which are essential and inessential for the trustworthiness of the result. The checking of the trustworthiness in the calculated results requires the detection of essential errors and the ignoring of inessential errors.

Safety-related systems receive initial data from sensors, i.e., measurement results, which are approximate data [15].

Therefore, from the beginning of the emergency mode, on-line testing changes the goal pursued in the normal mode to its characteristic checking of the trustworthiness in the result.

In normal mode, characterized by stability and duration, the safety-related system can operate at noise levels with little change in the input data received from the sensors. Such conditions lead to low checkability of digital circuits and contribute to the accumulation of hidden faults in a significant amount.

The start of the emergency mode is associated by an increase the activity of input data and their diversity. Such a multiple failure collapses the fault tolerance of the digital components and the functional safety of the critical system [5].

However, the described scenario, which is quite typical for modern safety-related systems, does not reveal another aspect underlying it and associated with evolutionary processes. These processes are explained by the resource-based approach, which analyzes the integration of models, methods and means of the computer world into the natural one [16, 17].

In the evolution of these resources, the approach identifies three levels: a) replication; b) diversification; c) self-sufficiency as the goal of development. At the level of replication, integration occurs in the absence of close limiting contact with the natural world, i.e., in resource niches: ecological, technological, market and others. In them, development is carried out by stamping clones, where the survival advantage is provided by an increase in productivity.

Filling the niches leads to their closure and dooms the clones to extinction. Survival is only possible through the manifestation of features that allow a level of diversification to be achieved. At this higher level, integration takes place in close contact with the natural world and is realized due to trustworthiness, i.e., adequacy to this world. Close contact is reflected in the processes of structuring resources for the peculiarities of the natural world.

The history of the development of computer resources has shown to the greatest extent two such features: parallelism and fuzziness, which, in particular, are demonstrated in the predominant hardware support for parallelization in the approximate data processing. All promising areas of development of modern technologies, including quantum computing, cyber-physical systems, the Internet of things and everything, as well as safety-related systems, are associated with the parallel processing of approximate data.

The computer world demonstrates all levels of resource evolution, but the priority is given to replication. Hardware solutions are stamped using matrix structures composed of their homogeneous elements. A distinctive feature of these structures, presented in parallel circuits of adders, shifters, multipliers and dividers, is the processing of data in parallel codes.

Software products are stamped by attaching bulky redundant modules with little use of their functionality [18]. This slagging of software is supported by the open performance and memory capacity of modern computer systems.

Matrix structures introduce into the development of computer resources the limitations inherent in replication as the lowest level of evolution.

The main purpose of matrix parallelism is to parallelize computations over time. However, parallel codes do not receive proper parallelization in data processing. Here and below the matrix architecture is analyzed using a matrix multiplier. This is due to the fact that multiplication is the main operation

of approximate data processing. Multiplication is used indirectly in the floating-point number notation itself. This leads to the fact that multiplication is actually used in all mantissa operations. The results of such operations have the same properties as product.

Binary n-bit codes are multiplied in one clock cycle on n^2 operational elements with a delay formed by the serial connection of $2n - 2$ elements.

Thus, each of the n^2 operational elements is used in a cycle only on its $(2n - 2)$ th part, that is, by 1.6% and 0.8% for $n = 32$ and $n = 64$, respectively [17].

The rest of the cycle is spent on parasitic transitions, which form the main part of the dynamic component of power consumption and are supplemented by a significant static component determined by the large size of the matrices. Truncated arithmetic operations reduce the manifestation of the described shortcomings in the processing of approximate data.

The development of critical systems shows the main disadvantage of matrix structures associated with the low checkability of FPGA components.

Typically, matrix structures use a wide variety of values accepted by parallel binary codes, to a negligible fraction. For example, the input word of the iterative array multiplier is formed from n bits of two operands and takes 2^{2n} values, which for $n = 64$ is 2^{128}. In this case, the operation of the multiplier even on a billion values uses only 2^{98}th part of their entire variety.

However, in the normal mode of critical systems, diversity can be limited to a few values of input words and result in a correspondingly low checkability of the digital components [17].

Analysis of the growth challenge allows you to determine the ways to solve it based on the improvement of components to the level of critical systems. The most radical solution is to reduce matrix structures and switch to parallelizing computations in sequential codes. However, today such a decision is not possible given the current realities.

The dominance of matrix structures over the past decades has led to the creation of a powerful infrastructure formed by computer resources to support the processing of parallel codes using matrix schemes.

Modern CAD systems, including FPGA design with LUT-oriented architecture, are also involved in the process of increasing the role of matrix structures. This process is supported not only by a matrix LUT-oriented architecture, but also by iterative array multipliers built into the microcircuit, schemes for accelerated addition of parallel codes, and access to extensive libraries of ready-made solutions with a matrix structure.

In these conditions, it is advisable to pull up the components to the level of critical systems by improving the matrix structures to the level of diversification, including the LUT-oriented architecture of FPGA components.

This architecture is organized in the form of a matrix of logical elements LE, containing a programmable one-bit register and a LUT unit, built on the basis of SRAM memory and a multiplexer composed of 2:1 switches connected in a pyramid scheme.

The LUT unit has Z inputs and 2^Z memory bits and is used as a logic function generator from Z variables. Calculations are implemented in FPGA architecture by decomposing them into logical functions.

In the process of programming an FPGA project, the description of these functions is entered into the memory of the LUT units in the form of a program code. The unit's LUT memory contains 16 bits for the most frequently used case $Z = 4$.

The memory is accessed at $dcba_2$, which is formed at inputs A, B, C and D using address bits a, b, c and d.

3. The essence of the proposed method

In FPGA components, faults can be identified with the distortion of the memory bits of the LUT units, which allows for a unified approach to checkability analysis in relation to various manifestations of faults in LUT-oriented circuits.

In a digital circuit based on LUT elementary units, the computational process is configured by means of program code from the LUT nodes memory.

The FPGA program code is verified to be correct using a checksum. This check makes all the bits written to the LUT memory checkable during normal FPGA mode. But the checkability of the LUT memory bits does not affect the faults associated with the distortion of these bits.

So, reading these bits can be performed by distortion, which is created by a faulty switch that selects the bits when they are read.

When assessing the testability of the LUT-oriented circuit, the possibility of constant faults in the switches that read bits of program code from memory is taken into account. These switches are based on a pyramid scheme, which has several levels. In the case of $Z = 4$, this pyramid contains 4 levels, in which 8, 4, 2 and 1 switches are located from inputs to outputs, respectively. At the first level, the read bit is fetched directly from the unit's LUT memory as a set of eight bits.

A fault occurring at any input of one of the switches of this level is identified with a fault of the read LUT memory bit. A similar fault of the next layer is identified with fault of two bits of the memory LUT, which can be selected by switches of this layer.

Reading any of these bits will overwrite their values with the switch fault value. Thus, these bits turn out to be associated with one fault, i.e., independently of each other, they indicate the existence of the same fault. Each next level doubles the number of bits associated with the same switch fault.

Since the identification of faults occurring in switches with faults in the memory LUT bits does not classify switches as checkable, it becomes necessary to study the checkability of the LUT-oriented circuit in terms of multiplexers.

A switch fault can manifest itself as an error only if the erroneous value does not match the bit value read from the memory LUT.

In addition, the switch output must be observed at the outputs of the LUT-oriented circuit. Then the switch becomes checkable with respect to this fault.

For the second and subsequent layers of the multiplexer circuit, where faults are identified with faults in several bits of the memory LUT, the checkability of switches of these layers increases with each associated bit.

In safety-related systems, the checkability of a LUT-oriented scheme receives additional restrictions imposed on the conditions for its evaluation. The priority position of the emergency mode differentiates the memory LUT bits and gives the bits observed in this mode a status important to ensure the functional safety of the critical system. These bits are assigned to the set *S*. Therefore, the checkability of the LUT-oriented circuit is considered and evaluated in relation to faults, the manifestation of which is identified with faults in the set of *S* bits.

Bits important to functional safety are diversified by dividing them into two sets of S_{CN} and S_{UN} bits, which are respectively checkable and non-checkable in normal mode.

Checkability can be extended by associated bits if the set A of these bits includes the sets $A_{CN} = A \cap S_{CN}$, $A_{CN} \neq \varnothing$ and $A_{UN} = A \cap S_{UN}$, $A_{UN} \neq \varnothing$ bits that are important to security and are respectively checkable and non-checkable in normal mode.

Until now, checkability has been evaluated taking into account only the bits of the A_{CN} set using the following formula:

$$C_O = B_{CN} / B_S, \tag{1}$$

where B_{CN} and B_S are the number of bits in the S_{CN} and S sets, which is calculated taking into account all LUT units of the LUT-oriented circuit.

The non-checkable bits A_{UN} associated with a single fault with the checkable bits of the A_{CN} set repeat the behavior of these bits with respect to the uniting fault, and therefore become indirectly also checkable. Indeed, a fault that is the same for the bits of the A_{CN} set and for the bits of the A_{UN} set will be observed in normal mode in the A_{CN} bits as well as in the A_{UN} bits with the start of the emergency mode. Conversely, if the bits of the A_{CN} set indicate no fault, then there will be no such fault in the A_{UN} bits in emergency mode. Thus, the checkability of the circuit is increased in the normal mode at the expense of the bits of the A_{UN} set that are not observed in this mode.

The bits of the set A_{UN} can be diversified into sets A_{U1} and A_{U2}, which are checkable to varying degrees depending on the values received by their bits.

If the bits of the set $A_{CN} \neq \varnothing$ take both values 0 and 1, then the bits of the set A_{UN} are checkable with respect to both values and 1 and 0 of the uniting stuck-at fault. Indeed, in one of the indicated value 0 or 1, the fault will necessarily manifest itself in the form of an error, i.e., it will become

detectable in the normal mode for on-line testing methods. Such sets A_{UN} are further considered as sets A_{U1}. If all the bits of the set $A_{CN} \neq \varnothing$ take only one value 0 or 1, then the bits of the set A_{UN} are checkable with respect to only the value that is inverse for the value determined by the uniting stuck-at fault. Indeed, the fault will not show itself if the value determined by it coincides with the values of all bits of the A_{CN} set and becomes detected in the normal mode for the bits of the A_{UN} set, otherwise. Such sets $_{AUN}$ are further considered as sets A_{U2}.

Taking into account the bits of the set A_{UN}, which become testable due to the uniting fault and the bits of the set $A_{CN} \neq \varnothing$, increases checkability, which can be estimated by the following formula:

$$C_A = C_O + (B_{U1} + B_{U2} / 2) / B_S, \quad (2)$$

where B_{U1} and B_{U2} are the number of bits in the sets A_{U1} and A_{U2}, which is calculated taking into account all LUT units of the LUT-oriented circuit.

Thus, the original checkability C_O is supplemented by the amount $G = (B_{U1} + B_{U2}) / B_S$, i.e., by its part $L = G / C_O$.

The method proposed in this paper is such a sequence of steps.

Step 1. Simulate of a LUT-oriented circuit on the normal and emergency input data and determine the S, S_{CN} and S_{UN} sets for the memory of each LUT unit.

Step 2. Modeling all the uniting stuck-at faults that can occur at the inputs of the switches, and defining the sets A, A_{CN} and A_{UN} for them in each LUT unit.

Step 3. Checking bit values in the A_{CN} sets and forming the A_{U1} and A_{U2} sets for each LUT unit.

Step 4. Counting the number of bits B_{CN}, B_S, B_{U1} and B_{U2} in the sets A_{CN}, S, A_{U1} and A_{U2}, taking into account all LUT units of the circuit. Determination of the original C_O and augmented C_A checkability according to formulas (1) and (2), as well as indicators G and L of the LUT-oriented scheme checkability extension.

4. Case Study

In order to test the proposed method, we developed software that implements this method. This software was developed in the environment Delphi 10 Seattle demo version [19].

The initial data of the developed program include a description of the LUT-oriented circuit with an indication of the number and list of LUT units, their program codes and the structure of the circuit. The program is also guided by data on LUT units of LE elements operating in arithmetic mode.

To implement the method, LUT units are ranked in order of closeness to circuit inputs. The input data are formed separately for normal and emergency modes of operation of the system. A threshold T is used for this separation. The input data values are then assigned to the circuit inputs. The developed software evaluates the checkability of the circuit at eight different values of the threshold, which are varied with a given step.

The operation of the circuit is simulated by performing computations at each LUT unit in the order of units in an ordered list. Calculations are performed by forming an address at the inputs of the LUT unit and reading a bit from the LUT memory at this address, taking into account the program code stored in it.

The read bit is transmitted from the output of the LUT unit to the inputs of the next LUT units, where a similar computational process takes place.

In the arithmetic mode of LE elements, the LUT unit uses the lower half of its program code to form a bit in the fast carry chain.

The simulation distinguishes between the normal and emergency addressable bits of the LUT memory. To form the sets S, S_{CN} and S_{UN}, the observability of each addressable bit is estimated by propagating its inverse value to the outputs of the circuit. The number of bits is counted in the set S.

For each stuck-at fault at the inputs of the switches in each LUT unit, the program generates a set of A associated bits, divided into checkable and non-checkable sets A_{CN} and A_{UN} in normal mode. The number of bits is counted in the A_{CN} set.

As a result of checking the values of the bits in the A_{CN} sets, the program generates the sets A_{U1} and A_{U2} for each LUT unit and counts the number of bits B_{U1} and B_{U2}.

Next, the program calculates checkability C_O and C_A and its expansion indices G and L for the LUT-oriented circuit.

The main program window is shown in Fig. 1 for the case of application of the checkability assessment method to the FPGA design of the iterative array multiplier of 8-bit binary codes from the LPM_mult CAD Quartus library.

FPGA project is implemented in the Intel Cyclone 10 LP FPGA: 10CL025YU256I7G [20]. The implementation was performed using CAD Quartus Prime 20.1 Lite Edition. The resulting LUT-oriented circuit, built using 101 LUT units, has 16 inputs, 16 outputs. 16 LUT units are connected to the outputs of the component.

The simulation starts and the program is exited using the "Start" and "Exit" keys. Key "P: 3 - 24" shows the values of the threshold P, which in this case takes 8 values from 3 to 24 and changes in steps of 3. In normal mode, the factors take on values less than the threshold.

In emergency mode, the circuit processes the rest of the input data. The next key "LUT # 6" allows to view the memory of the LUT unit 6 for all values of the threshold P. Pressing this key increases the number of the considered LUT unit with a step of 1 in a circle. Inputs A, B, C and D used in the considered LUT unit 6 are shown on the right: "Used inputs: A B C D".

The memory of the LUT unit is shown as a square matrix of bits, the numbers of which are made up of the numbers of rows and columns: 00_2, 01_2, 10_2, 11_2. The rows and columns of the matrix are indicated by specifying two pairs of inputs D, C and B, A, on which the $dcba_2$ address of selected bit is formed. Bits of S_{CN} and S_{UN}, which are checkable and non-checkable in normal mode, are colored blue and yellow, respectively.

As the threshold P increases, the amount and variety of input data in the normal mode increases, resulting in non-checkable bits moving from the set of S_{UN} to the set of S_{CN}.

The original bit ratio 14 : 2 in these sets changes to a ratio 12 : 4.

In the first level of switches, their faults are identified with faults in individual bits of the LUT memory. Faults of the second-level switches are identified with pairs of associated bits located at adjacent addresses 0000_2 and 0001_2, 0010_2 and 0011_2, 0100_2 and 0101_2, etc.

Level 3 switches bind the bits located in the rows of the LUT memory. Thus, each next level connects pairs of the *A* sets of the previous level.

At the bottom of the panel, simulation results are shown for various thresholds P.

Lines B_{CN}, B_{UN}, B_{U1} and B_{U2} show the change in the number of checkable and non-checkable bits in the A_{CN}, A_{UN} sets, as well as checkable bits in the A_{U1} and A_{U2} sets, complementing the original checkability C_O.

An increase in the threshold P leads to an increase in the number of checkable bits of the A_{CN} set from 253 to 850 and a decrease in the number of non-checkable bits from 992 to 475.

As a result of this bit reallocation, the C_O checkability increases from 20.3% to 64.1%.

Bits of the sets A_{U1} and A_{U2} allow to get augmented C_A checkability, which increases from 37.1% to 79.8%.

The indicators of its increase relative to the initial *C*O checkability decrease in absolute terms from 16.8% to 15.7% and in relative terms from 82.7% to 24.4%.

Thus, the addition of checkability increases with a decrease in the activity of the input data, when checkability is most in demand, and reaches 82.7% with the lowest threshold P = 3.

5. Conclusions

Checkability plays just as important a role in ensuring functional safety as the fault tolerance of a circuit, ensuring its transformation into fail safety. Safety-related systems can operate in two modes: normal and emergency. In each of these modes, the system circuits show significantly different levels of checkability.

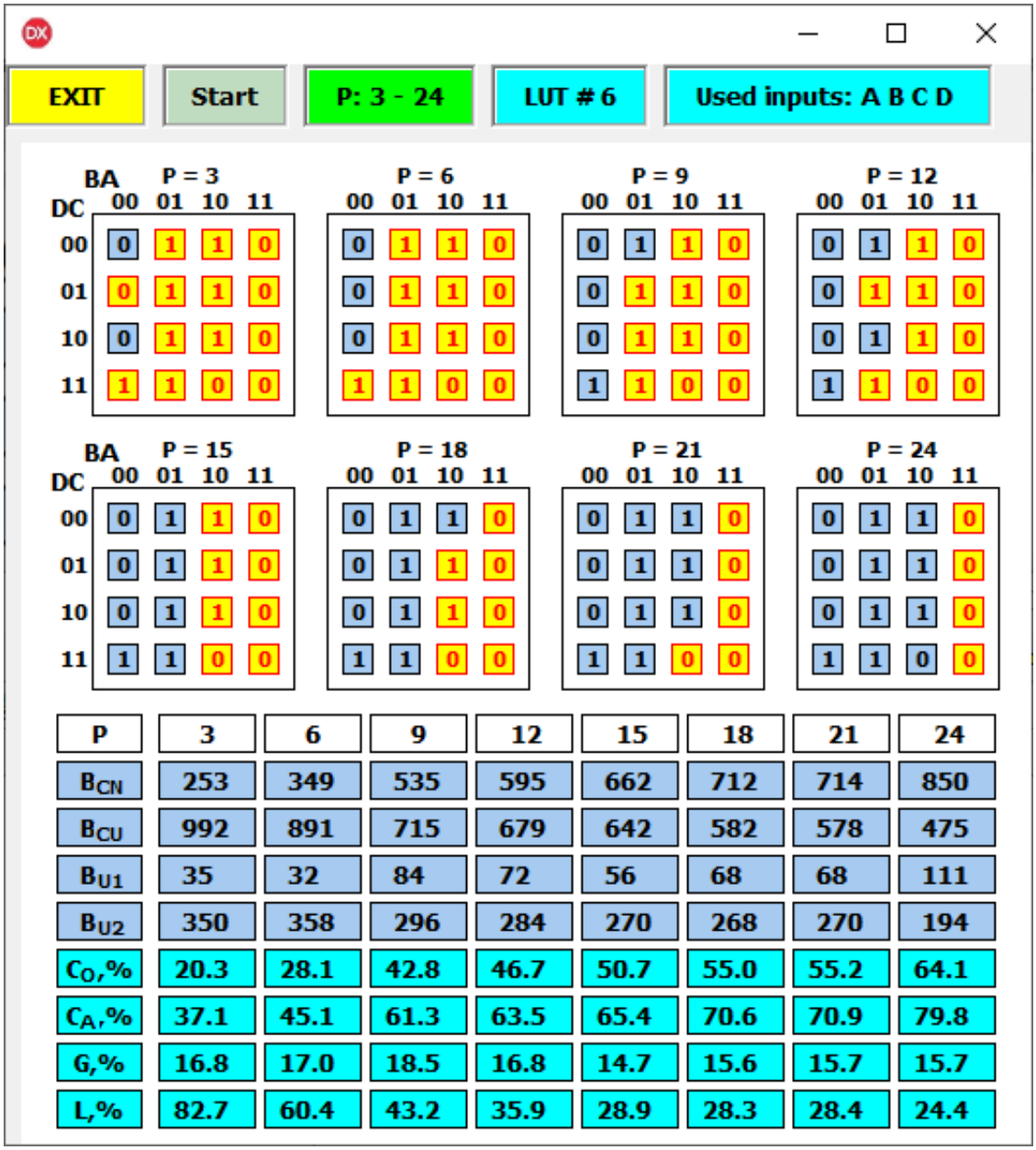
In normal mode there is a low activity of changing input data. This results in low checkability over a long period of time. This can result in latent faults that do not show up in normal mode. In emergency mode, the input change activity increases greatly. As a result, the checkability of the circuits increases and hidden faults become apparent. Thus, different levels of checkability in each mode cause multiple failures. This threatens the fault tolerance of critical system components.

LUT-oriented circuits of FPGA components developed for safety-related systems allow their faults to be identified with faults in bits of the LUT memory. This approach does not allow extending the checkability of LUT memory to all faults of LUT-oriented circuits, but creates a unified basis for analyzing their influence on the checkability of the circuit.

The memory LUT bits, which determine the results computed in emergency mode, are referred to as bits important to functional safety. These bits are differentiated by dividing them into two sets: bits that are checkable and non-checkable in normal mode. The ratio of the number of checkable bits to all bits important to functional safety determines the checkability of LUT-oriented circuits.

The proposed method complements this checkability assessment by analyzing the memory LUT bits associated with a single fault.

Due to a common fault, the non-checkable bits can be observed in normal mode with the associated checkable bits and thus also become checkable. Experiments have shown an 82.7% increase in the checkability of LUT-oriented 8-bit iterative array multiplier circuits under the most critical conditions of the lowest activity of the normal mode input data.



P	3	6	9	12	15	18	21	24
B_{CN}	253	349	535	595	662	712	714	850
B_{CU}	992	891	715	679	642	582	578	475
B_{U1}	35	32	84	72	56	68	68	111
B_{U2}	350	358	296	284	270	268	270	194
C_O,%	20.3	28.1	42.8	46.7	50.7	55.0	55.2	64.1
C_A,%	37.1	45.1	61.3	63.5	65.4	70.6	70.9	79.8
G,%	16.8	17.0	18.5	16.8	14.7	15.6	15.7	15.7
L,%	82.7	60.4	43.2	35.9	28.9	28.3	28.4	24.4

Figure 1: Main panel of the program implementing proposed method

6. References

[1] V. Kumar, L.K. Singh, A.K. Tripathi, P. Singh, Safety analysis of safety critical systems using state space models, IEEE Software 34 (2017) 38–47.

[2] IAEA Safety Standards, Specific Safety Guide No.SSG-39, Design of Instrumentation and Control Systems for Nuclear Power Plants, No. SSG-39 Specific Safety Guide, 2016.

[3] Siamak Alizadeh, Srinivas Sriramula, Impact of common cause failure on reliability performance of redundant safety related systems subject to process demand, Reliability Engineering & System Safety 172 (2018) 129–150.

[4] J. Kim, E. S. Kim, J. Yoo, Y. J., J. G. Lee Choi, An Integrated Software Testing Framework for FPGA-Based Controllers in Nuclear Power Plants, Nuclear Engineering and Technology 48(2) (2016) 470–481.

[5] O. Drozd, V. Romankevich, M. Kuznietsov, M. Drozd, O. Martynyuk, Using natural version redundancy of FPGA projects in area of critical applications, in: Proceedings of the 11th IEEE International Conference on Dependable Systems, Services and Technologies, Kyiv, Ukraine, 2020, pp. 58–63. DOI: 10.1109/DESSERT50317.2020.9125050.

[6] Hafizul Asad, Ilir Gashi, Diversity in Open Source Intrusion Detection Systems, Computer Safety, Reliability, and Security, Lecture Notes in Computer Science 11093 (2018) 267–281.

[7] 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.

[8] Henrik Eriksson, Andreas Soderberg, Andreas Soderberg, Remote Sensing and Functional Safety, in: Proceedings of the 2th ITSITS European Congress, Strasbourg, France, 2017.

[9] H. Waidyasooriya, M. Hariyama, K. Uchiyama, Design of FPGA-Based Computing Systems with OpenCL. Springer, Cham, Switzerland, 2018. DOI: 10.1007/978-3-319-68161-0.

[10] N. Arya, A. P. Singh, Fault Tolerant System for Embedded System, International Journal of Engineering and Technology 99 (3S) (2017) 93–97.

[11] T. Hovorushchenko, Information Technology for Assurance of Veracity of Quality Information in the Software Requirements Specification. Advances in Intelligent Systems and Computing 689 (2018) 166–185.

[12] T. Hovorushchenko, A. Herts, Ye. Hnatchuk, Concept of Intelligent Decision Support System in the Legal Regulation of the Surrogate Motherhood, CEUR-WS 2488 (2019) 57–68.

[13] Lokesh Kamble, Prachi Palsodkar, Prasanna Palsodkar, Research trends in development of floating-point computer arithmetic, in: Proceedings of the International Conference on Communication and Signal Processing (ICCSP), 2017.

[14] Ronak Shah, Glitch Analysis and Reduction in Combinational Circuits, Computer Science & Information Technology (2016) 33–40.

[15] L. Chen, J. Han, W. Liu, and F. Lombardi, On the Design of Approximate Restoring Dividers for Error-Tolerant Applications, IEEE Transactions on Computers 65(8) (2016).

[16] J. de Fine Licht, M. Blott, T. Hoefler, Designing Scalable FPGA Architectures Using High-Level Synthesis, in: Proceedings of 23nd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, Vienna, Austria, 2018.

[17] H. Amano, Principles and Structures of FPGAs, Springer, USA, New-York, 2018.

[18] M. Ebrahimi, R. Sadeghi, Z. Navabi, LUT Input Reordering to Reduce Aging Impact on FPGA LUTs, IEEE Transactions on Computers 69(10) (2020) 1500–1506.

[19] Delphi 10 Seattle: Embarcadero, 2022. URL: https://www.embarcadero.com/docs/datasheet.pdf.

[20] Intel Cyclone 10 LP Device, 2022. URL: https://www.intel.com/content/dam/support/us/en/programmable/kdb/pdfs/literature/hb/cyclone-10/c10lp-51001.pdf.