

Diagnosis of Input Variable Faults of Logical Functions Using Test Map

VLADIMIR HAHANOV¹, VOLODYMYR OBRIZAN¹, SVETLANA CHUMACHENKO¹,
EUGENIA LITVINOVA², HANNA KHAKHANOVA¹, TETIANA BORYSENKO³

¹Design Automation Department, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine

²Department of Computer Aided Design, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine

³Department of Information Control Systems, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine
Kharkiv National University of Radio Electronics, Kharkiv, Ukraine

Corresponding author: Vladimir Hahanov (e-mail: hahanov@icloud.com).

ABSTRACT This paper introduces a vector-logical mechanism for intelligent computing designed to diagnose non-functional components in models and hardware–software systems. It focuses on detecting faults, defects, destructive modules, embedded trojans, and application activations that produce functional “noise” unrelated to core business processes. The proposed mechanism relies on harmonic interaction between model and algorithm, where the model’s exponential redundancy allows algorithmic nullification of faulty behavior. Two key components of in-memory intelligent computing architecture – modelling and simulation – are analyzed for diagnostic and prognostic tasks. Deterministic models are proposed to perform big-data analysis and classification using a similarity–difference metric, with a truth table serving as an exponentially redundant data structure for energy- and time-efficient processing. An analytical method for solving the covering problem with quadratic computational complexity is presented to generate minimal fault-covering test sets. The proposed method diagnoses all faults in the input variables of RTL, System, and Gate-Level functionality. The study concludes with recommendations for applying this prompt-computing diagnostic approach to improve the reliability and verification of models, software, and hardware systems, supported by practical cloud-based examples using the MOSI service.

KEYWORDS in-memory computing, vector logic, fault modeling, circuit diagnosis, simulation modeling.

I. STATE OF THE ART

Intelligent computing is defined as *modeling for simulation* to determine inference, which includes diagnosis, prognosis, or their derivatives. Diagnostic computing is the process of determining the technical state of an object or its condition with an accuracy ranging from a fully functional state to any combination of multiple faulty states. This work focuses exclusively on deterministic vector-logical in-memory *modeling for simulation* mechanisms (see Fig. 1) for solving tasks related to the diagnosis of functional deviations. Modeling and simulation maintain a harmonic relationship that allows the computational complexity of the algorithm to be reduced to zero by leveraging the model’s exponential redundancy. The vector-logical computing mechanisms for diagnostics address the following tasks: 1) Identification of faults in hardware–software models based on a test map or a deductive matrix; 2) Detection of destructive trojans and malicious codes in hardware–software systems using a deductive matrix; 3) Determination of similarity–difference tests between two or more functionalities using an XOR matrix; 4) Equivalence mapping of faults for classification or clustering

based on an XOR matrix; 5) Automatic generation of a minimal test for a specified class of model or object faults using a test map; 6) Detection of non-functional (noise) applications and screens during the execution of business processes within a company via an XOR matrix; 7) Construction of fault testing schemes for critical variables using a deductive matrix; 8) Project or object verification based on vector-logical *testbench* modeling as a secondary model. The vector-logical mechanisms exhibit a systemic nature, remaining invariant with respect to the types of diagnostic objects and the kinds of faults or defects encountered.

The level of development in computer engineering depends directly on the maturity of technologies that address *design and test* challenges. This dependency is well understood: to release a new product to the market, it is essential to have robust mechanisms for model verification and tools for diagnosing the product during operation. Therefore, the mechanisms, models, and algorithms for *design and test* must be several steps ahead of fabrication technologies [1–8] and must exhibit intelligent capabilities [4]. In practice, however, this is rarely the case. Typically, new fabrication technologies for a product are

introduced first [1, 5, 8], and only afterward do scientists and engineers develop new mechanisms for testing, verification, and diagnostics [9–16]. This creates an inherent contradiction: we can design virtually anything in terms of scale, but it is not always possible to fully verify the entire design. Standards have been established [9–12] for partitioning large projects into segments (IP cores) that are more manageable for modern verification and diagnostic tools. However, the existing mechanisms (models and algorithms) for verifying and diagnosing IP cores remain far from optimal in terms of automation and time–energy efficiency metrics [11–13]. Alternative graphs are used for diagnosing RTL models in which faults are not tied to the structure of the diagnostic object [3]. Challenges such as automatic *testbench* generation, fault modeling without explicit algorithms, and fully automated defect diagnostics for digital products have yet to be resolved, despite the emergence of promising solutions in the literature [17–20]. Unfortunately, these published mechanisms are typically local in scope [21–27] and cannot be regarded as invariant with respect to the wide variety of diagnostic object types.

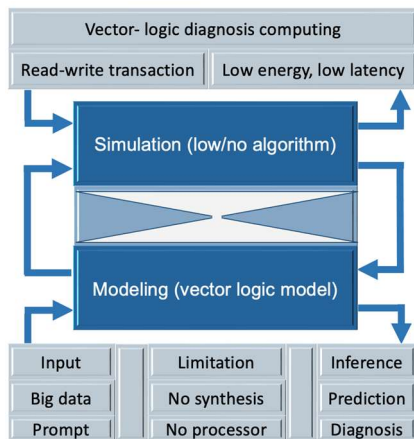


Figure 1. Vector-logical computing for diagnostics

The main idea of the proposed research is to develop simple vector-logical mechanisms that are invariant with respect to the types of verification and diagnostic objects [28–31]. The metric of such in-memory mechanisms must meet the criteria of time–energy efficiency and ensure linear complexity of simulation algorithms by leveraging the exponential redundancy of memory allocated for modeling procedures [1, 28, 29].

The objective of this research is to reduce computing time and energy consumption while improving the quality of digital products through the implementation of verification and diagnostic mechanisms based on vector-logical theory.

The specific tasks are as follows: (1) development of in-memory mechanisms for vector-logical verification and diagnostics of digital functionalities and circuits; (2) development of mechanisms for testing and diagnosing destructive codes and redundant (noise) data within a company’s business processes; (3) verification of the proposed mechanisms across various logical functionalities represented by vector-logical models.

II. DIAGNOSIS COMPUTING

Diagnosis aims to identify the location and type of faults in a circuit that prevent the computing device from operating according to its intended function. Modeling for simulation

refers to the preparation of data (testbench) for conducting a diagnostic experiment on a model or an actual device. A testbench is the mechanism for organizing the diagnostic experiment; it includes the testing algorithm and the model in the form of an input test and the circuit’s output responses (UUT) to the test. The outcome of circuit testing is a logical vector resulting from an XOR comparison of the circuit’s output states and the testbench model at the XOR gate output (Response vector), where unit values indicate the presence of defects or faults in the circuit (see Fig. 2). The diagnostic algorithm is the procedure for decoding the response vector of the XOR element’s state to determine the location and type of defects based on the test map in the actual device or its model. A test map is the simplest model for locating logical defects in a functionality defined by a logical vector.

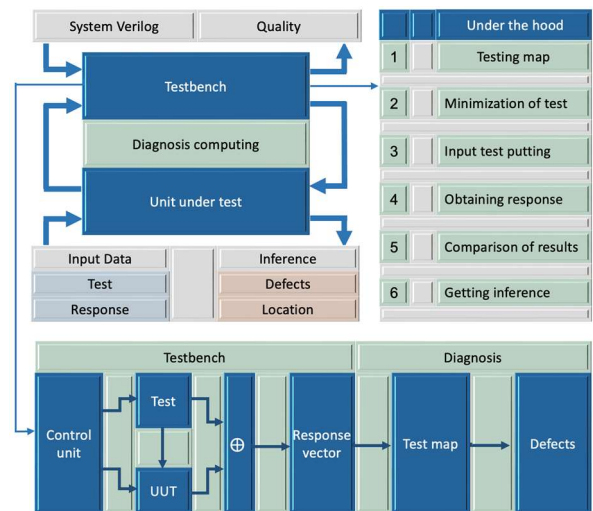


Figure 2. Architecture of diagnosis computing

Several algorithms are proposed for diagnosing input defects in functionalities based on the test map:

(1) intersecting (or merging) the rows (test sets) of the test map where an incorrect output state of the functionality is recorded, to diagnose single (or multiple) input logical faults;

(2) intersecting (or merging) the rows (test sets) of the test table where an incorrect circuit output state is recorded, to diagnose single (or multiple) logical faults in the circuit lines;

(3) constructing a conjunction of disjunctive terms using the 1-coordinates of the selected rows of the deductive matrix, where the logical product of all conjunctive terms in CNF yields a disjunctive normal form (DNF) representing all possible combinations of detectable faults for the given test sets;

(4) identifying faulty columns in the fault table by the active coordinates of the deductive matrix, which correspond to input test sets with incorrect output states, followed by refining the unit values of coordinates in the selected fault table columns using inverted values of the corresponding input coordinates of the test sets, and performing coordinate-wise union (for multiple fault detection) and intersection (for single fault detection) operations on the selected columns of the fault truth table to diagnose the maximal and minimal sets of faults in the input variables.

III. COMPUTING OF DESTRUCTIVE CODES

Diagnosis aims to identify the location and type of destructive codes (functionalities) in a computer that prevent the

computing device from operating according to its intended purpose. Modeling for simulation refers to the preparation of data (sandbox) for conducting a diagnostic experiment on both a malware-infected model and a clean device. A sandbox (see Fig. 3) is the mechanism for organizing the test-based diagnostic experiment; it includes testing algorithms and malware models in the form of an input test and the circuit's output responses (UUT) to the test. The result of the test is a signature or a logical vector obtained by XOR comparison of the output states of the clean and malware-infected computer at the XOR gate output (signature vector), where a non-zero signature (malware-ID) indicates the presence of destructive elements in the computer. The diagnostic algorithm is the procedure for decoding the response vector (signature) of the XOR element's state to determine the location and type of malware based on a test map or a library of known destructive codes. A test map is the simplest model for detecting malware in the functionalities of a computing device, defined by a logical vector. Several algorithms are proposed for diagnosing malware functionalities based on the test map.

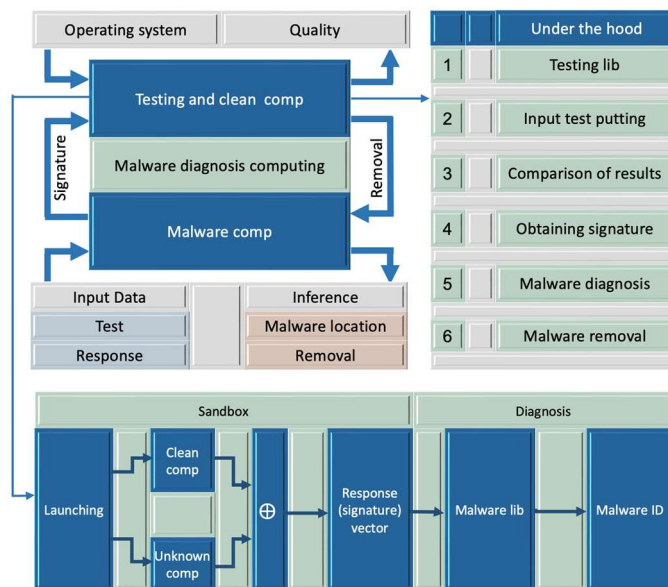


Figure 3. Computing of destructive codes

IV. COMPUTING NON-FUNCTIONAL DATA IN A BUSINESS PROCESS

An Autonomous agent is an active modeling system for screen fields, treated as test sets that an operator must complete to minimize functionality execution time and identify destructive (redundant) screens that slow down the operator (or client). A system that requires a developer for routine support will not be marketable in today's IT landscape. Modern demand is for prompt-engineering systems that can provide a comprehensive, user-satisfying response to any brief query (prompt). To achieve this, the system must perform a true function modeling procedure using information from the screens. Implementing such a system in Python or Java code is feasible for a team of three specialists within six months. Practically any functionality involving the completion of active screen fields can be mapped to a truth table of 5–8 variables. Verification or testing of such functionality is easily performed using a test map constructed from the logical vector representing the functionality (see Fig. 4). The agent operates in two modes: (1) activation of screen fields for the operator's or client's use; (2)

diagnostic mode for identifying redundant screens within already executed functionalities, along with generating an operator or client activity map.

Computing within an IT company can be summarized in two points: Modeling (true function) for Simulation. If a company attempts to design a more complex system, it will either (1) never be completed or (2) fail to sell on the market. A system must be as simple as a brick. However, the principles of modern computing theory must not be ignored — this is a fundamental rule for competent managers.

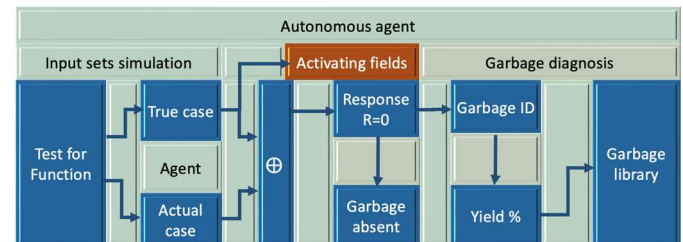


Figure 4. Computing for non-functional data diagnosis

V. SIMILARITY–DIFFERENCE IN DIAGNOSIS

Similarity–Difference. Knowing the 1-coordinates of the $\oplus$ -matrix (Difference) implies that all remaining coordinates are Similarity. The converse is also true, since Similarity is formed by unit values derived from the $(= \oplus)$ relation — that is, equality of ones (11) and zeros (00). There is no observed dependence between the cardinality of Similarity–Difference coordinates in a vector and the cardinality of Similarity–Difference coordinates in the Cartesian XOR matrix (see Fig. 5).

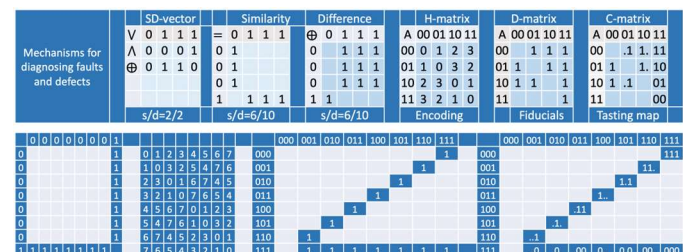


Figure 5. Mechanisms for diagnosing faults and defects

To construct a test that distinguishes one functionality from another, it is necessary to build a Difference matrix for these functions. The unit coordinates of the resulting matrix indicate the test sets required to differentiate a true function L from a false function F. Thus, the following conclusion can be drawn: to build a difference test between two functionalities, the logical vectors L and F of these functionalities must be available. This fundamental theory underpins the testing and diagnostics of destructive codes in computer functionalities. A functionality difference test has an important property — there always exists at least one test set that distinguishes the first functionality from the second. The drawback, however, is that the logical vector F of the second functionality must be known. What should be done if only one functionality L is present? In this case, an XOR matrix of the logical vector L should be constructed, where the unit coordinates of the matrix define all reference points of this functionality and the corresponding test sets to verify these points. If, during the diagnostic experiment

on these test sets, different states are recorded, then these points of mismatch can be used to construct and identify the functionality F that distorts the given function L . This is a distinct and practically significant task for diagnosing destructive software codes and hardware trojans. A straightforward solution involves modeling both functionalities — the destructive F and the true L — to construct a test map for determining a diagnostic test for the destructive functionality.

VI. SUPERPOSITION OF THE DEDUCTIVE MATRIX WITH THE TRUTH TABLE

Objective: To enhance the performance and quality of fault modeling in logic circuits through the superposition of the truth table for single and multiple constant input faults with the deductive vector matrix. Tasks: (1) develop an input fault model to evaluate test quality; (2) superimpose the truth table with the deductive vector matrix to analyze test quality; (3) design an algorithm for the combined analysis of the deductive vector matrix and the truth table for single and multiple constant input faults. Input data: (1) the logical vector of the functionality Q (see Fig. 6) and the binary test set x -test; (2) derived data — the deductive vector matrix D and the truth table X for single and multiple constant input faults.

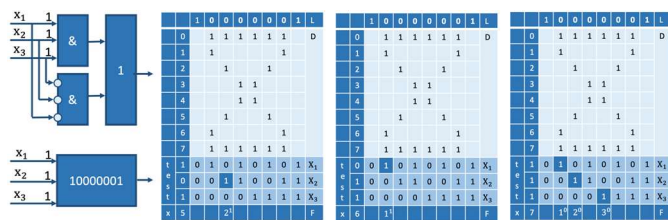


Figure 6. Superposition of the deductive matrix and the truth table

This section presents the superposition of two matrices of identical dimensions: the deductive vector matrix and the truth table encompassing all combinations of constant input faults for the logic element shown on the left in two forms: (1) term implementation in DNF; (2) an element based on a logical vector. The columns of the truth table X use unit coordinates to represent exhaustive combinations of single and multiple constant faults derived from the binary input test set. The symbol $X_{ij} = 1$ denotes a single fault $\bar{x}_i$. A multiple fault is identified by multiple 1-coordinates within a column of the matrix X . For example, in the blue cells, unit entries indicate faults that are tested by the input test sets $x = 101$ (2^1), 011 (1^1), and 111 ($1^0, 2^0, 3^0$). The algorithm for forming the coordinates of the vector of detectable faults involves a single logical multiplication operator $D_x: L = X \wedge D_x$, where the deductive vector D_x (for $x = 5, 6, 7$) is treated as the output state vector of the truth table for the detectable faults under the binary input test sets $101, 011$, and 111 . The result of this multiplication, through its unit coordinates, defines the set of detectable faults, which are identified in the last rows of the tables using standard notation.

VII. DEDUCTIVE MATRICES IS A SET OF VECTOR DERIVATIVES

Two matrix operations define the algorithmic complexity for obtaining all deductive vectors and all Boolean derivatives of logic of any complexity. These are two structurally critical

components for diagnostics combined within a single matrix, derived using the logical vector of the functionality. A column of the deductive matrix (the vector form of a Boolean derivative) is a logical vector that excludes active variables when converted into disjunctive normal form, yet retains the conditions under which this variable activity is possible. The XOR function does not require any conditions for variable activation; the Boolean derivative with respect to any variable of this function is equal to one. The deductive matrix has two key properties: (1) the row set represents vectors for transporting input faults to the output of the logic element; (2) the column set consists of Boolean derivatives for all combinations of input variables. Each Boolean derivative is a complete set of logical conditions, defined over the exhaustive test set, for activating a specific combination of input variables. These vector conditions are used to construct a minimal test (see Fig. 7). Such a test must cover all ones in the deductive matrix corresponding to the four columns (first-order Boolean derivatives) that contain a single unit in the address codes of the upper truth table. An exception is the XOR function, where the minimal test comprises two mutually exclusive sets.

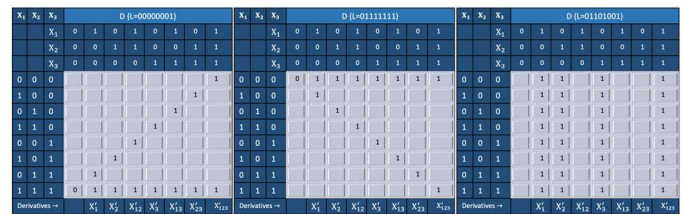


Figure 7. Deductive vectors and Boolean derivatives

VIII. DIAGNOSIS CIRCUITS SYNTHESIS OF THE INPUT VARIABLE FAULT

In critical systems, it is essential to have online diagnostics for any faults that may occur in digital control circuits. Such circuits must be very simple both in operation and in fault diagnosis. Therefore, the deductive matrix, with its exponentially redundant memory for storage, serves as an ideal model for solving the problem of diagnosing input variable faults. The simplest solution involves two logic elements (see Fig. 8). The first is the standard functionality, represented by the logical vector L . The second element comprises four columns of the deductive matrix, which represent the Boolean derivatives with respect to four output variables. A Boolean derivative defines the conditions for transporting any fault of the given input to the output. This scheme is fully sufficient to perform real-time diagnostics of any fault or defect that may arise at the inputs of the functionality during the operation of the control circuit for a critical object.

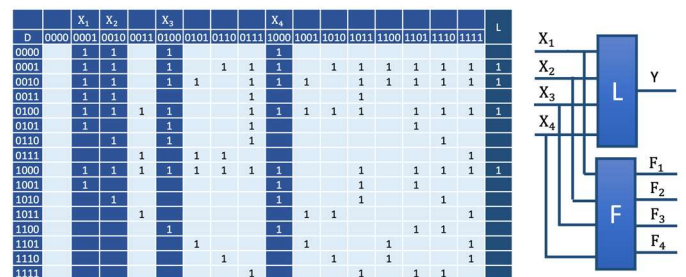


Figure 8. Synthesis of a diagnosis circuit using two elements

A more advanced version of the input fault diagnostic mechanism for a critical object control unit is implemented using four circuits synthesized by deriving minimal disjunctive normal forms. Karnaugh maps are employed for this purpose, enabling the generation of minimal analytical expressions (see Fig. 9). Each expression excludes the variable with respect to which the Boolean derivative is taken. This diagnostic approach offers the advantage that if one circuit fails, the remaining circuits continue to operate in normal mode.

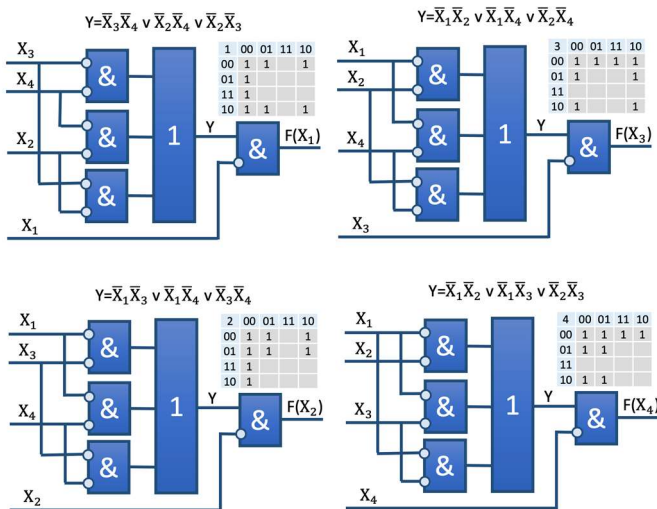


Figure 9. Synthesis of diagnostic circuits using minimal DNF

IX. DEDUCTIVE MATRIX AS A DIAGNOSIS MECHANISM

The deductive matrix is not only a collection of deductive vectors for fault modeling of logic elements within a digital circuit but also comprises logical vector-columns representing Boolean derivatives that ensure the transport of input faults under an exhaustive test. This feature can be leveraged to design diagnostic circuits for digital control systems of critical objects. A diagnostic circuit for input faults, based on logical vectors and their derivatives, serves as an engineering template that solves any diagnostic task by storing the appropriate derivative logical vectors in memory. Presented here are four modeling examples of functionality test maps (see Fig. 10), from which diagnostic circuits for input faults can be easily derived based on the derivative columns of the deductive matrix. The entire deductive matrix may also be used as a diagnostic model for single and multiple input faults. In this case, the diagnostic model comprises two elements: the first is the logical vector of the functionality, and the second is the deductive matrix itself. Such a diagnostic circuit can identify any defect formed by a combination of input faults in the given functionality.

[illegible]

Figure 10. Deductive matrix as a diagnosis mechanism

X. COMPLETE DIAGNOSIS SCHEME FOR LOGICAL FUNCTIONALITY

A diagnosis scheme for input faults based on logical vectors and their derivatives is presented. Synthesis refers to implementing the model within a technology-compliant system of logic elements for subsequent simulation of input sets, without requiring an explicit algorithm. Modeling refers to implementing the model in any memory, enabling the simulation of input sets through read–write transactions without processor involvement. Considering that memory and ALUs now offer comparable performance, synthesis is no longer the optimal solution for model implementation. Therefore, the focus shifts to modeling circuits that utilize logical vectors stored in memory to diagnose any input faults in digital functionalities. This approach is superior to synthesizing a logic circuit based on a disjunctive normal form derived from the unit values of the logical vector. Notably, a test map for any functionality can be generated from the logical vector in just three matrix operations (see Fig. 11).

Activity Matrix	Encod matrices	Deductive Matrix								Testing Map							
		000	001	010	011	100	101	110	111	000	001	010	011	100	101	110	111
1 1 1 1 0 0 0 1	0 1 2 3 4 5 6 7	000				1	1	1	1	000				.11	1.1	1.1	
1 1 1 1 1 1 1 1	1 3 2 5 4 6 7 6	001			1		1	1	1	010			.1		1.0	1.1	
1 1 1 1 1 1 1 1	2 3 0 1 1 7 4 5	010		1					1	100					1.0	1.1	110
0 1 1 1 1 1 1 1	1 3 2 1 0 7 6 5 4	011		1	1	1	1	1		101			.0		0.0	1.	
0 1 1 1 1 1 1 1	1 4 5 6 7 0 1 2 3	100				1	1	1	1	110				.11	0.1	0.1	0.1
0 1 1 1 1 1 1 1	1 5 4 7 6 1 0 3 2	101			1		1	1	1	111			.1		0.0	0.0	010
0 1 1 1 1 1 1 1	1 6 4 5 2 3 0 1 7	110							1	110			.1		0.1		00.010
0 1 1 1 1 1 1 1	7 6 5 4 3 2 1 0	111		1	1	1	1	1	1	111			.0		.0.0	.0.0	

Figure 11. Deductive matrix for the logical vector 11100001

In this process, the intermediate result is a deductive matrix whose vectors are used to model faults in the logic elements of a digital structure. But this is not all: the columns of the deductive matrix represent derivatives, which are themselves logical vectors and have the remarkable property of transmitting all signals — including faults — from the external input to the element's output. Considering that the deductive matrix contains 2^n columns (and rows), representing all possible combinations of derivatives with respect to the input variables, this discussion focuses only on the n columns corresponding to the number of functionality variables (see Fig. 12). Specifically, the columns representing the derivatives with respect to the input variables are ($X'_1 = 11100001$, $X'_2 = 00110011$, $X'_3 = 11111111$). These columns constitute three elements stored in memory, which will handle the detection and diagnosis of faults for the respective inputs.

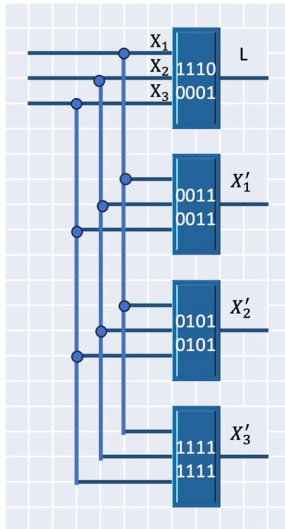


Figure 12. Fault diagnostic scheme for input variables

Given that the useful functionality includes a logical vector of the same length, the diagnostic scheme exhibits a 300% redundancy. Such overhead can only be justified by the necessity to diagnose and rectify faults in contexts where failures are extremely costly for society — foremost among them critical systems such as transportation, nuclear power, and avionics. It should be noted that Boolean derivatives are obtained as an intermediate product during the construction of the test map. If one were to use an analytical expression for computing the Boolean derivative, it would require a complex mathematical apparatus, historically employed decades ago for test generation (an NP problem). This further confirms that by exploiting the exponential redundancy of logical vectors, it is possible to derive algorithms for testing and modeling digital systems with linear computational complexity. This describes an engineering approach to solving complex testing and diagnostic tasks. Its core idea is to build an exhaustive test from the functionality's logical vector using three matrix operations, and then extract from this test any input sequences that adequately verify the significant faults. Naturally, the three diagnostic elements in the presented scheme can be combined into a single element with three outputs, corresponding to the number of first-order Boolean derivatives. If a derivative column consists entirely of ones, this indicates that an input (3) fault will be detected on all test sets.

A more general case: a diagnostic scheme for any combination of input faults, constructed using the deductive matrix of a functionality defined by the logical vector 10000001, is shown in Fig. 13.

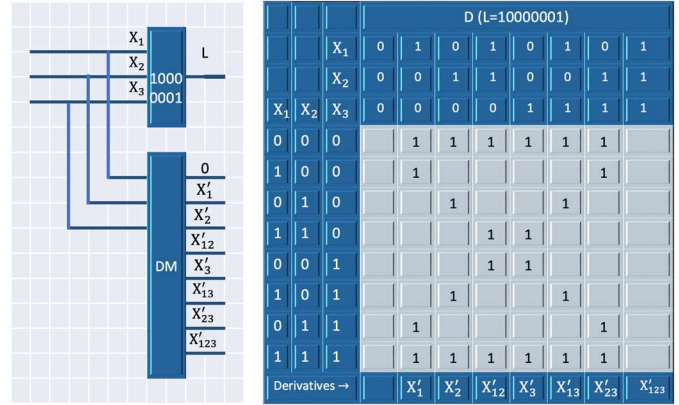


Figure 13. Diagnosis scheme for any combination of faults

Naturally, the number of outputs of the second element (DM) is determined by the columns of the deductive matrix. Depending on the specific tasks, one can select anywhere from a single column to the complete set of columns in the deductive matrix. For the given functionality, there will be six significant columns, which define six inputs for the DM element. The first and last outputs do not participate in diagnosing combinations of input faults. This is evident from the deductive matrix, where these two columns contain zeros in all coordinates. It should be noted that the diagnosis scheme is a flexible mechanism: it is also possible to use a single output that aggregates all derivatives or columns of the deductive matrix. Ultimately, this depends on the creativity and ingenuity of each designer and the requirements of the specific diagnostic task. In practice, it is generally sufficient to limit the number of outputs in the DM element to the number of inputs of the useful functionality.

XI. FAULT DIAGNOSIS BASED ON THE TEST MAP

The interaction of diagnosis mechanisms with the test map of the digital functionality is shown in Fig. 14.

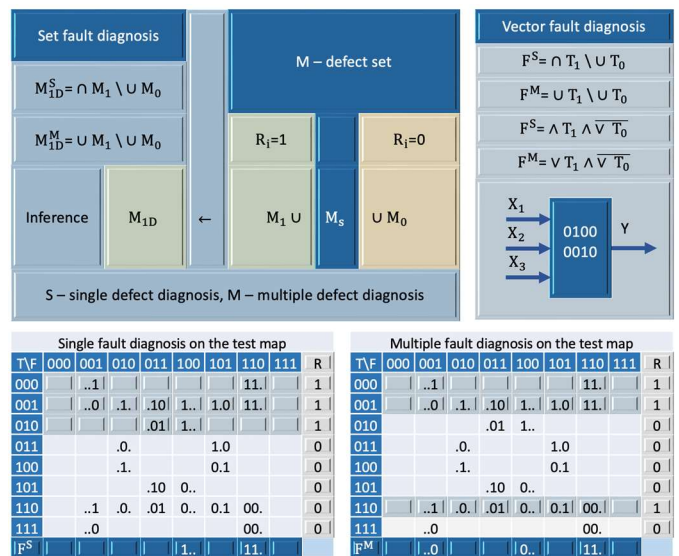


Figure 14. Fault diagnosis based on the test map

M_S represents the common subset of defects obtained through the set-theoretic intersection $M_0 \cap M_1$, whose subtraction allows a significant reduction in the defect set when establishing a diagnosis. To achieve this, the complete defect set is divided into two subsets: those detected ($R_i=1$) and those

not detected ($R_i=0$) by the input sets in the exhaustive test. Then, two search algorithms are applied for identifying single ($M_{1D}^S = \cap M_1 \setminus \cup M_0$) and multiple ($M_{1D}^M = \cup M_1 \setminus \cup M_0$) constant faults. The difference between the algorithms lies in whether they perform an intersection $\cap M_1$ or a union $\cup M_1$ of the unit rows in the test map according to the experimental verification vector. The first procedure, following Boolean algebra rules, forms a diagnosis for single faults: $M_{1D}^S(F^S = \wedge T_1 \wedge \overline{\vee T_0})$. The second procedure, also using Boolean algebra rules, forms a diagnosis for multiple faults: $M_{1D}^M(F^M = \vee T_1 \wedge \overline{\vee T_0})$. The lower part of Fig. 15 shows the faults M_{1D}^S and M_{1D}^M identified by these two procedures. To execute the diagnostic algorithm, it is necessary to form the experimental verification vector $R = \{0,1\}$ for all test sets and superimpose this vector with the test map.

XII. MODELING A MINIMAL FAULT COVERAGE TEST

A test map represents the complete system of relationships between an exhaustive test and all combinations of faults, defined by the XOR product of the logical vector (truth table). The test map can be used exclusively for tasks related to minimizing tests for specified faults or minimizing the faults detected by a given set of test inputs. An exact solution to both tasks can be obtained by constructing a minimal test that covers the specified fault set or by finding the minimal fault coverage detectable by a specified set of test inputs. Formally, the task reduces to writing disjunctive terms for the columns (rows) of the fault table and subsequently converting them into conjunctive terms in a disjunctive normal form (DNF), which encodes all combinations of minimal tests (or faults). How can the procedure for obtaining the DNF from the CNF be defined using vectors of the reduced test map? The test map, through three procedures ($M1-M2-M3$) and the experimental verification vector, is transformed into a table of detectable defects, which includes only the essential test rows that verify the faults that produced incorrect output states (see Fig. 15).

M1	1				0				M2	1				0				M3	1				0			
	4	3	2	1	4	3	2	1		1	2	3	4	5	6	7	8		1	2	3	4	5	6	7	8
0000	1		1						0000	1		1					1	0000	1		1					1
0001		1				1			0001		1			1			2	0001					1			2
0010					1				0010									0011				1		1		3
0011					1		1		0011				1		1		3	0100		1			1			4
0100		1		1			1		0100		1		1			1	4	1000	1	1						5
0101		1		1					0101									1001			1		1			6
0110					1		1		0110									1010						1		7
0111					1		1		0111									1101		1					1	8
1000		1		1					1000		1		1				5	Three procedures for modeling a compact fault table to solve the coverage problem								
1001					1				1001				1		1		6									
1010						1		1	1010						1		7									
1011								1	1011																	
1100			1						1100																	
1101			1				1	1	1101			1				1	1		8							
1110						1		1	1110																	
1111						1		1	1111																	

Three procedures for modeling a compact fault table to solve the coverage problem

Figure 15. Modeling the fault table for solving the coverage problem

The formation of CNF terms from the columns of the final table M3, followed by their conversion into DNF terms, is represented by the following analytical expressions:

$$\begin{aligned}
 T &= (1v5)(2v4v5v8)(1v6)(3v4)(2v6)(3v7)(4v8)(7v8)= \\
 &= (1v5)(1v6)(3v4)(3v7)(4v8)(7v8)(2v6)(2v4v5v8)= \\
 &= (1v56)(3v47)(47v8)(2v62v64v65v68)= \\
 &= (1v56)(38v47)(2v64v65v68)=
 \end{aligned}$$

$$\begin{aligned}
 &= (138v147v3568v4567)(2v64v65v68)= \\
 &= 1238v13468v13568v1368v1247v1467v14567v14678v \\
 &23568v34568v3568v3568v24567v4567v4567v45678= \\
 &= 1238v1368v1247v1467v3568v4567.
 \end{aligned}$$

Here, all possible options for covering the specified faults with all combinations of test sets are presented. This task is the search for a minimal fault coverage test.

The formation of CNF terms from the rows of the final matrix M3, followed by their conversion into DNF terms, is represented by the following analytical expressions:

$$\begin{aligned}
 F &= (1v3)(2v5)(4v6)(2v4v7)(1v2)(3v5)(6v8)(2v7v8)= \\
 &= (12v15v32v35)(13v15v23v25)(6v48)(2v7v48)= \\
 &= (12v15v32v35)(13v15v23v25)(62v67v48)= \\
 &= (123v125v123v125v135v15v1235v125v \\
 &v123v1235v23v235v135v135v235v235)(62v67v48)= \\
 &= (15v23)(62v67v48)= \\
 &= (1562v1567v1548v2362v2367v2348)= \\
 &= 1256v1567v1458v236v2348.
 \end{aligned}$$

Here, all possible options for covering the specified test sets with all combinations of detected faults are presented. This task is the search for a minimal set of faults that cover the given test sets.

XIII. DISCUSSION OF THE RESULTS

Generating the fault table M3 is invariant with respect to the form of the simulation results, whether from a circuit or a logical functionality. If the simulation results for a digital circuit are available (see Fig. 16), it is straightforward to construct a compact fault table M3 from them. Thereafter, the coverage problem can be solved analytically using the described procedures to minimize the exhaustive test for single faults across the 12 circuit lines.

The method has the following limitation: it does not process sequential circuits or elements with memory (counters, registers, triggers).

Table of test detected circuit fault												
Test sets	Quality		Circuit line fault detected									
	Q.set	Q.test	1	2	3	4	5	6	7	8	9	10
00000	0.12	0.12				1			0			0
00001	0.21	0.21				1		0			1	0
00010	0.25	0.46			0			0	1		0	1
00011	0.25	0.46			0			0	1		0	1
00100	0.12	0.46				1			0			0
00101	0.21	0.46				1		0			1	0
00110	0.29	0.50		1		0		0	1		0	1
00111	0.29	0.50		1		0		0	1		0	1
01000	0.12	0.50				1			0			0
01001	0.25	0.54	1			1		0			1	0
01010	0.29	0.58			1	0			0	1	0	1
01011	0.29	0.58			1	0			0	1	0	1
01100	0.08	0.58							0			0
01101	0.08	0.58							0			0
01110	0.21	0.71		0	0				1	0		0
01111	0.21	0.71		0	0				1	0		0
10000	0.12	0.71				1			0			0
10001	0.25	0.71		1		1		0			1	0
10010	0.25	0.71			0			0	1		0	1
10011	0.25	0.71			0			0	1		0	1
10100	0.12	0.71				1			0			0
10101	0.21	0.71				1		0			1	0
10110	0.29	0.71		1		0			0	1	0	1
10111	0.29	0.71		1		0			0	1	0	1
11000	0.25	0.83				1	1		0	0		1
11001	0.42	1.00	0	0	1		0	1		1	0	1
11010	0.29	1.00			1	0			0	1		1
11011	0.21	1.00			1				0		0	1
11100	0.17	1.00							0	0		1
11101	0.25	1.00			0				1	0		1
11110	0.29	1.00		0	0				1	0		1
11111	0.29	1.00		0	0				1	0		1

Figure 16. Simulation results of single faults in the circuit

XIV. ANALYTICAL SOLUTION TO THE COVERAGE PROBLEM

The CNF constructed from the M3 columns limits the number of row combinations for the minimal coverage algorithm by transforming the CNF into DNF. The analytical algorithm for constructing the minimal coverage involves the following steps:

1. Formation of disjunctive terms of the conjunctive normal form based on the unit coordinates of the M3 matrix columns.
2. Logical multiplication of all conjunctive normal form terms to generate conjunctive terms of the disjunctive normal form, thereby obtaining the complete solution set (see Fig. 17).

M3	1	2	3	4	Disjunctive Normal Form Coverage
1	1	1			$T=(1v3) (1v4) (2v4) (2v3)=$
2			1	1	$(11v14v13v34) (22v24v23v34)=$
3	1			1	$= (1v34) (2v34)=$
4		1	1		$=12v34.$

Figure 17. Analytical solution to the coverage problem

3. After each iteration of multiplication, it is necessary to perform minimization within parentheses of the set of conjunctive terms in the disjunctive normal form using the rules of elimination and idempotency: $aVab=a$, $aa=a$.

4. The resulting DNF terms after transformation contain minimal covers (with the minimal number of literal symbols per term), which correspond to minimal tests. The algorithm for obtaining the minimal cover has quadratic computational complexity related to transforming CNF into DNF and subsequent minimization of DNF terms.

5. Naturally, there exists a simpler vector-logical algorithm for constructing minimal covers, but its discovery is a matter of time.

XV. CONCLUSION

Diagnosis computing is a straightforward vector-logical mechanism for identifying combinations of faults detected by exhaustive or minimal tests. Several vector-logical diagnostic computing mechanisms are proposed, utilizing the deductive matrix to construct logic circuits for online fault diagnosis during device operation. A vector-logical mechanism for intelligent computing is introduced to diagnose non-functional components of models and hardware–software products. Faults and defects, destructive components and trojans, as well as the activation of applications generating functional “noise” unrelated to business process execution, are examined. Vector-logical mechanisms and architectures for testing and diagnosing faults and defects in logical functionalities are discussed. The mechanism represents a harmonic relationship between the model and algorithm, enabling algorithm nullification through the model’s exponential redundancy. Two components of an in-memory intelligent computing architecture, modeling for simulation, are considered to address diagnostic and prognostic tasks. Deterministic models for intelligent computing are proposed for analyzing big data to classify it according to a similarity–difference metric. This utilizes the truth table as an exponentially redundant data structure model, capable of solving all relevant market tasks efficiently in terms of time and energy, within an in-memory vector-logical computing framework. An analytical method for solving the coverage problem is presented, featuring quadratic computational complexity to derive a minimal coverage test for

specified faults from the full set of input vectors. The advantage and main difference from known methods is that the proposed method diagnoses all faults in the input variables of RTL, System, and Gate-Level functionality. Recommendations are provided for applying the proposed prompt computing diagnostics to engineers, students, educators, and managers to enhance the effectiveness of testing and verification of models, software, and hardware products. Examples of applying prompt computing for diagnosing faults and defects based on the modeling for simulation mechanism are also presented.

References

- [1] V. Hahanov, *Cyber-Physical Computing for IoT-driven Services*, New York: Springer, 2018. 279p. <https://doi.org/10.1007/978-3-319-54825-8>.
- [2] M. Mánik and E. Gramatová, “Efficient diagnostics algorithms for regular computing structures,” *Proceedings of the 14th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems*, Cottbus, Germany, 2011, pp. 87-92, <https://doi.org/10.1109/DDECS.2011.5783054>.
- [3] R. Ubar, J. Raik, M. Jenihhin, A. Jutman, *Structural Decision Diagrams in Digital Test: Theory and Applications*, Birkhäuser Cham, Computer Science Foundations and Applied Logic Series, 1st ed., 2024, XIII+595 pp., ISBN 978-3-031-44733-4 (hardcover), ISBN 978-3-031-44736-5 (softcover), ISBN 978-3-031-44734-1 (eBook). <https://doi.org/10.1007/978-3-031-44734-1>.
- [4] S. Zhu, T. Yu, T. Xu, H. Chen, S. Dustdar, S. Gigan, D. Gunduz, E. Hossain, Y. Jin, F. Lin, B. Liu, Z. Wan, J. Zhang, Z. Zhao, W. Zhu, Z. Chen, T. S. Durrani, H. Wang, J. Wu, T. Zhang, Y. Pan, “Intelligent computing: The latest advances, challenges, and future,” *Intelligent Computing*, vol. 2, article ID: 0006, 2023. <https://doi.org/10.34133/icomputing.0006>.
- [5] M. Abdulkasoud, N. Dehadrai, J. Castrillón, A. Sakr and R. Schuster, “Edge diagnostics platform: Orchestration and diagnosis model for edge computing infrastructure,” *Proceedings of the 2021 IEEE International Conference on Edge Computing (EDGE)*, Chicago, IL, USA, 2021, pp. 51-59, <https://doi.org/10.1109/EDGE53862.2021.00017>.
- [6] H. R. H. Al-Absi, A. Abdullah and M. I. Hassan, “Soft computing in medical diagnostic applications: A short review,” *Proceedings of the 2011 National Postgraduate Conference*, Perak, Malaysia, 2011, pp. 1-5, <https://doi.org/10.1109/NatPC.2011.6136288>.
- [7] S. Kaur and E. K. Lee, “Diagnostic analysis: Directional relation graph,” *Proceedings of the 2019 IEEE International Conference on Cluster Computing (CLUSTER)*, Albuquerque, NM, USA, 2019, pp. 1-5, <https://doi.org/10.1109/CLUSTER.2019.8891032>.
- [8] J. Xu, S. Guo, H. Guo and X. Tian, “A novel diagnostic method for single and dual power switch open-circuit faults of six-phase FTPMSM system even in fault tolerant operation,” *IEEE Transactions on Power Electronics*, vol. 37, no. 8, pp. 9777-9789, 2022, <https://doi.org/10.1109/TPEL.2022.3161437>.
- [9] X. Zhao, P. Jiang and L. Liu, “Research of parallel scheduling strategy for hierarchical SiP test using IEEE 1500 standard,” *Proceedings of the 2014 15th International Conference on Electronic Packaging Technology*, Chengdu, China, 2014, pp. 1108-1111, <https://doi.org/10.1109/ICEPT.2014.6922838>.
- [10] G. Ali, F. A. Hussin, N. B. Z. Ali and N. H. Hamid, “Enhancement in IEEE 1500 standard for at-speed functional testing,” *Proceedings of the 2014 5th International Conference on Intelligent and Advanced Systems (ICIAS)*, Kuala Lumpur, Malaysia, 2014, pp. 1-5, <https://doi.org/10.1109/ICIAS.2014.6869507>.
- [11] P. Inglese, E.-I. Vatajelu and G. Di Natale, “Side channel and fault analyses on memristor-based logic in-memory,” *IEEE Design & Test*, vol. 41, no. 3, pp. 29-35, 2024, <https://doi.org/10.1109/MDAT.2023.3324522>.
- [12] D. Gajaria, K. A. Gomez and T. Adegbija, “STT-RAM-based hierarchical in-memory computing,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 35, no. 9, pp. 1615-1629, 2024, <https://doi.org/10.1109/TPDS.2024.3430853>.
- [13] M. Zhang, A. Lebeck and D. Sorin, “Fractal consistency: Architecting the memory system to facilitate verification,” *IEEE Computer Architecture Letters*, vol. 9, no. 2, pp. 61-64, 2010, <https://doi.org/10.1109/L-CA.2010.18>.
- [14] R. Ben Fekih, M. Lahami, S. Bradai and M. Jmaiel, “Formal verification of ERC-based smart contracts: A systematic literature review,” *IEEE Access*, vol. 13, pp. 11396-11422, 2025, <https://doi.org/10.1109/ACCESS.2025.3527158>.

- [15] W. Chen, S. Ray, J. Bhadra, M. Abadir and L.-C. Wang, "Challenges and trends in modern SoC design verification," *IEEE Design & Test*, vol. 34, no. 5, pp. 7-22, 2017, <https://doi.org/10.1109/MDAT.2017.2735383>.
- [16] A. Kamput, C. Dechsupa, W. Vatanawood and S. Pomsiri, "Scalable timed-automata models for traffic light control systems: Challenges and solutions in formal verification," *IEEE Access*, vol. 12, pp. 124260-124281, 2024, <https://doi.org/10.1109/ACCESS.2024.3455097>.
- [17] I. Pomeranz, "Computation of seeds for LFSR-based diagnostic test generation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 12, pp. 2004-2012, 2015, <https://doi.org/10.1109/TCAD.2015.2459031>.
- [18] I. Pomeranz, "Diagnostic test generation that addresses diagnostic holes," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 2, pp. 335-344, 2019, <https://doi.org/10.1109/TCAD.2018.2812121>.
- [19] I. Pomeranz, S. M. Reddy and S. Venkataraman, "z-Diagnosis: A framework for diagnostic fault simulation and test generation utilizing subsets of outputs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 9, pp. 1700-1712, 2007, <https://doi.org/10.1109/TCAD.2007.895758>.
- [20] I. Pomeranz, "Low-power diagnostic test sets for transition faults based on functional broadside tests," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 11, pp. 2428-2432, 2014, <https://doi.org/10.1109/TVLSI.2013.2290768>.
- [21] X. Wu, R. Tian, S. Cheng, T. Chen and L. Tong, "A nonintrusive diagnostic method for open-circuit faults of locomotive inverters based on output current trajectory," *IEEE Transactions on Power Electronics*, vol. 33, no. 5, pp. 4328-4341, 2018, <https://doi.org/10.1109/TPEL.2017.2711598>.
- [22] J. Liu et al., "Open-circuit fault diagnosis strategy for five-phase permanent magnet fault-tolerant servo motor drive systems based on phase voltage," *IEEE Transactions on Industry Applications*, vol. 61, no. 3, pp. 3612-3622, 2025, <https://doi.org/10.1109/TIA.2025.3532912>.
- [23] S. Zhou, Z. Chen and T. Lin, "Lithium-ion battery cell open circuit fault diagnostics: methods, analysis, and comparison," *IEEE Transactions on Power Electronics*, vol. 38, no. 2, pp. 2493-2505, 2023, <https://doi.org/10.1109/TPEL.2022.3211568>.
- [24] Y. Hu, S. Cheng, X. Wu, C. Xiang and Z. Li, "A diagnostic method for open-circuit faults of loads and semiconductors in 3L-NPC inverters," *IEEE Journal of Emerging and Selected Topics in Power Electronics*, vol. 11, no. 3, pp. 2577-2590, 2023, <https://doi.org/10.1109/JESTPE.2022.3205897>.
- [25] I. Pomeranz, "RETRO: Reintroducing tests for improved reverse order fault simulation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 8, pp. 1930-1934, 2020, <https://doi.org/10.1109/TVLSI.2020.2997762>.
- [26] P. Wang, A. M. Gharehbaghi and M. Fujita, "An automatic test pattern generation method for multiple stuck-at faults by incrementally extending the test patterns," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2990-2999, 2020, <https://doi.org/10.1109/TCAD.2019.2957364>.
- [27] L. Anghel, R. Cantoro, R. Masante, M. Portolan, S. Sartoni and M. S. Reorda, "Self-test library generation for in-field test of path delay faults," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 11, pp. 4246-4259, 2023, <https://doi.org/10.1109/TCAD.2023.3268210>.
- [28] V. Hahanov et al., "Prompt-testing of logic," *Proceedings of the 2024 IEEE East-West Design & Test Symposium (EWDTS)*, Yerevan, Armenia, 2024, pp. 1-5, <https://doi.org/10.1109/EWDTS63723.2024.10873774>.
- [29] D. Devadze et al., "In-memory fault-free vector simulation," *Proceedings of the 2023 IEEE East-West Design & Test Symposium (EWDTS)*, Batumi, Georgia, 2023, pp. 1-6, <https://doi.org/10.1109/EWDTS59469.2023.10297076>.
- [30] V. Hahanov, V. Obrizan, I. Hahanova and E. Fomina, "Verification of digital system by a new asserting mechanism based on IEEE 1500 sect standard," *Proceedings of the International Conference Mixed Design of Integrated Circuits and System, MIXDES 2006*, Gdynia, Poland, 2006, pp. 544-548, <https://doi.org/10.1109/MIXDES.2006.1706639>.
- [31] Y. Zheng, G. Shcherbakova, B. Rusyn, A. Sachenko, N. Volkova, I. Kliushnikov, S. Antoshchuk, "Wavelet transform cluster analysis of UAV images for sustainable development of smart regions due to inspecting transport infrastructure," *Sustainability*, vol. 17, issue 3, 927, 2025, <https://doi.org/10.3390/su17030927>.



Vladimir Hahanov Professor of the Design Automation Department at Kharkiv National University of Radio Electronics, Doctor of Technical Sciences, Professor, IEEE Senior Member, Head of the specialized scientific council.



Volodymyr Obrizan Senior Lecturer at the Design Automation Department at Kharkiv National University of Radio Electronics, PhD in Computer Engineering



Svetlana Chumachenko Head of Design Automation Department at Kharkiv National University of Radio Electronics, Member of EMC, Member of STC, IEEE Senior Member, Doctor of Technical Sciences, Professor



Eugenia Litvinova Professor of the Department of Computer Aided Design of Computers at Kharkiv National University of Radio Electronics, Scientific Secretary of the Specialized Scientific Council, Dr. of Science, Professor



Hanna Khakhanova Professor of the Design Automation Department, Computer Engineering Faculty, Kharkiv National University of Radioelectronics, Doctor of Technical Sciences, Professor



Tetiana Borysenko PhD, Associate Professor, Associate Professor Department of Information Control Systems Kharkiv National University of Radio Electronics, Ukraine, tetiana.borysenko@nure.ua, ORCID: 0000-0001-6915-6861