МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ ТА ОБЧИСЛЮВАЛЬНІ МЕТОДИ

DOI
УДК 61.326:519.713

W. Gharibi [1], PhD, Prof., **A. Hahanova** [2], Cand. T. Sc, Ass. Prof.,
V. Hahanov [2], D.Sc., Prof., **S. Chumachenko** [2], D.Sc., Prof.,
E. Litvinova [2], D.Sc., Prof., **I. Hahanov** [2]
[1] The University of Missouri-Kansas City, MO 64110 USA,
gharibiw2002@yahoo.com
[2] Kharkiv National University of Radio Electronics,
Ukraine, 61166, Kharkiv, Nauka Avenue, 14,
(057) 7021 326, hahanov@icloud.com

Vector-deductive Memory-based Transactions for Fault-as-address Simulation

The main idea is to create logic-free vector computing, using only read-write transactions on address memory. The strategic goal is to create a deterministic vector-quantum computing using photons for read-write transactions on stable subatomic memory elements. The main task is to implement new vector computing models and methods based on primitive read-write transactions in vector flexible interpretive fault modeling and simulation technology, where data is used as addresses for processing the data itself. The essence of vector computing is read-write transactions on vector data structures in address memory. Vector computing is a computational process based on elementary read-write transactions over cells of binary vectors that are stored in address memory and form a functionality where the input data to be processed is the addresses of these cells. The advantages of a vector universal model for a compact description of ordered processes, phenomena, functions, and structures are defined for the purpose of their parallel analysis. Analytical expressions of logic, which require algorithmically complex calculators, are replaced by output state vectors of elements and digital circuits, focused on the parallelism of register logical procedures on regular data structures. A vector-deductive method for formula synthesis for propagating input lists (data) of faults is proposed, which has a quadratic computational complexity of register operations. A new matrix of deductive vectors has been synthesized, which is characterized by the following properties: compactness, parallel data processing based on a single read-write transaction in memory, elimination of traditional logic from fault simulation procedures, full automation of its synthesis process, and focus on technological solving all problems of technical diagnosis. In the work, the transition to vector logic in the organization of computing and the elimination of traditional logic presented in the form of tables and analytical expressions were carried out. The use of read-write transactions on memory in the absence of a command system focuses the new vector-logic computing towards deterministic quantum architectures based on stable subatomic memory particles.

K e y w o r d s: vector computing, vector form of logic, matrix of deductive vectors, table of synthesis of deductive vectors, analytical form of structures, deductive-vector method of fault analysis, digital circuit, vector model of defects, functions and structures, sequencer of vector deductive fault simulation.

The motivation for the study is dictated by several factors of the following metrics: quantum computing of combinatorial problems based on qubit data structures, the use of elementary read-write transactions on memory-driven computing with a vector description of logic, memory redundancy for storing interpretive flexible logic models, the use of data as addresses for increasing performance of deductive simulation.

In design and test, three main forms of describing processes and phenomena are used: tabular, analytical, graph [1—6]. In this case, the matrix (table) and the vector are two forms for describing models that pass into each other. A vector (binary, multi-valued) is a compact form of a truth table in the form of an ordered sequence of output states, if the input address components are sorted in ascending order [7—13]. The matrix, if necessary, expands into a one-dimensional vector for the convenience of parallel data processing in register memory. Naturally, it is enough to simply restore the table or matrix from the vector form of the description of the process or phenomenon.

Statement 1. Further, a vector appears as the simplest form of describing a function and structure, which is focused on placement in address memory.

Quantum errors [14—15] are the cost to pay for efficient quantum computations of combinatorial problems. Inside a quantum computer, information has unique properties compared to data in standard microelectronics. The hardware of a quantum computer is more than a billion trillion times more likely to fail than a classical computer. This enormous susceptibility to error is the single biggest problem preventing quantum computing from realizing its claimed potential.

An approach known as quantum error correction (QEC) can solve this problem, at least in principle. But these experiments have not yet reached the level of quality required to reduce the overall error rate in the system. Even in a fault-tolerant architecture, in both classical and quantum computers, additional complexity generates new paths to error. The current correction estimates or break-even threshold for quantum hardware is approximately 1 error in 1000 operations. I would like to bring this indicator to the level of at least 5 errors per 1000 operations.

Statement 2. Redundancy is always good for classical and quantum computing for reliable, error-free operation. The vector form of describing processes and phenomena is good for both quantum and classical computing. The paradigm of creating a quantum market computer using quantum logic based on qubit data structures is questionable from a market perspective.

Metric bases are the following:

• *Q*-logic, like the usual one, creates up to 90 percent of all problems related to the technology of its hardware implementation, performance, quality, reliability, testing, diagnostics, and repair.

• The exclusion of classical and quantum logic from the computer makes it possible to move to a higher level of computing architecture, which uses only regular memory for storing and logical data transformation.

• At the same time, the shortcomings associated with the use of ALU and data buses that are mandatory today, which slow down computing processes tenfold, are eliminated.

• The architecture of a *Q*-computer is becoming promising, using only the structure of electrons or crystal lattices of low-dimensional quasiparticles [16] as a memory and transactions between them based on controlled photons.

• The speed of accessing the electronic memory and performing transactions on it is determined by the speed of light, which makes such *Q*-computer attractive in the market and unique for solving combinatorial problems.

• The technical and temperature conditions for the implementation of a quantum computer are still an insurmountable constraint for its mass implementation in the existing cyberspace.

Several approaches have been proposed [17—19] to testing (BIST), diagnosing and repairing memory structures based on redundancy, as well as for constructing and evaluating redundancy analysis algorithms based on preference vectors for memory devices with spare elements. Experiments on the use of new algorithms for self-testing and repair (STAR) of SRAM type memory have shown the effectiveness of the proposed approach. There is a trend: on regular memory for processing big data, all design and test, diagnosis and repair tasks are technologically solved more simply than on a classic computer that uses a standard ALU-based processor.

The bottleneck of a classical computer is the communication channel between a slow memory and a fast logical processor. At present, memory performance has increased tenfold, so the communication channel with the logical processor is becoming an unacceptable obstacle to improving the performance of computer architecture. Solution: transfer all calculations to memory, remove the logical processor from the computer architecture, at the same time solving the problem of bottlenecks in the interaction between memory and the logical processor [12, 19].

Statement 3. Switch to memory-driven computing, where the function of logic should be performed by memory elements with vectors written in it. Such computing is as close as possible in terms of data structures to quantum parallel computing, but not on quantum logic, which creates big problems, but on read-write transactions in quantum memory.

Efficient fault simulation [20—22] for combinational circuits have been known for several decades. However, sequential failure simulation, which is often used in testing and fault tolerance applications, remains a very time-consuming problem, especially for large circuits.

A new deductive method is proposed for error simulation at the RTL level and on the system model of high-level decision diagrams. Simulation acceleration is achieved through efficient data structures implemented to perform many operations in the deductive fault simulation algorithm. Experiments on RTL reference circuits show that this method achieves up to two orders of magnitude reduction in operating time compared to failure simulation at the gate level. Raimund Ubar singles out deductive simulation as the most elegant mathematical apparatus that has good prospects for processing complex digital circuits.

Statement 4. The implementation of the deductive method on vector data structures placed in memory has no analogues in terms of processing speed of complex digital elements due to significant memory redundancy. The method has the potential to be implemented in digital and quantum circuits for fault simulation, test quality assessment, error diagnosis and elimination.

Most of the 615 papers by Irith Pomeranz and co-authors are devoted to generating tests for combinational and sequential circuits, creating efficient data structures for algorithms of design and test, fault simulation and diagnostics. A few examples are considered below. They are determining the structural similarity of circuit parts and then applying almost deductive methods to generate tests for each of the parts; generation of additional setup tests to clarify the diagnosis when using the feedback shift register; a specialized format for storing data about failures on the tester.

A hybrid approach is proposed in which for some tests, full output responses are stored, while for other tests, only pass/fail information is stored. The issues of simulating various types of faults that exist in modern microelectronics are considered. Emphasis is placed on compacting and compressing data, tests, faults, and diagnostic information based on the introduction of an undefined $x = (0,1)$ state to denote the flexibility of a function.

Statement 5. Data compaction due to the redundancy of the alphabet of variables is a very important property for technical diagnosis, along with the flexibility of data homogeneity, the ability to use them to describe various types of faults in modern digital devices. All these properties fit into the vector form of representation of logical functions and faults.

Next, models and methods for testing and simulating digital systems are proposed, containing certain advantages of quantum computing in terms of the implementation of vector qubit data structures in the technology of classical computing processes. It is known that data structures are decisive for the crea-

tion of efficient parallel algorithms and high-performance computing devices. Therefore, the development of mathematically perfect and technologically simple data structures in smart companies takes about 80 percent of the design time, when about 20 percent of time and material resources are spent on algorithms and their hardware-software coding. A simple conclusion follows from this – it is necessary to find such primitives of data structures that will significantly simplify parallel high-performance algorithms running on them.

Parallelism in solving combinatorial problems is a physical property of quantum computing, which in classical computing for parallel fault simulation and fault diagnosis is provided by unitarily encoded data structures due to redundant memory [7, 11, 16, 18]. The topic is of particular interest in the field of design and test from researchers involved in testing [1] and verification of digital systems-on-chips in the emerging technology of nanoelectronics.

The subatomic level of detailing chips requires the development of new approaches in the field of test synthesis, failure simulation and diagnosis to create testable and fault-tolerant computing systems [6, 14, 15, 16—19]. This is confirmed by numerous publications in the design and test technology segment. This topic is supported by 253,359 publications in the IEEE Xplore library, of which fault modeling accounts for about 20% (47,227) of published papers. In addition, a corresponding assessment of the two mentioned lines of research was published by Springer, where 182,717 and 15,603 monographs and articles were published, respectively. The classification of fault simulation technologies can be represented by the relationship of data structures, models, methods and types of defects (Fig. 1) [8].

Statement 6. The main problems in the field of fault simulation should be defined as follows:

• High computational complexity of combinatorial analysis algorithms at the register level of model description [2, 4—7, 14—22].

• The complexity of the algorithms for modeling and simulating sequential circuits associated with an unpredictable number of iterations [1—6, 20—22].

• A significant amount of data structures for the analysis of digital systems-on-chips, which negatively affects the performance of fault simulation methods and test synthesis [1, 3, 6, 9, 16—19].

•The complexity of the analysis and synthesis of tests for high-dimensional logic and their structural and combinatorial complexity associated with converging fan-outs [1—6, 7—9, 23—25].

• Parallel solution of the problems of testing and simulating digital devices [1, 16—19, 20—22].

Thus, the crisis of modern computing is associated with two problems:

1) the processing of big data by an insolvent processor-memory pair;

2) a catastrophic increase in power consumption by global computing processes in modern microelectronics.

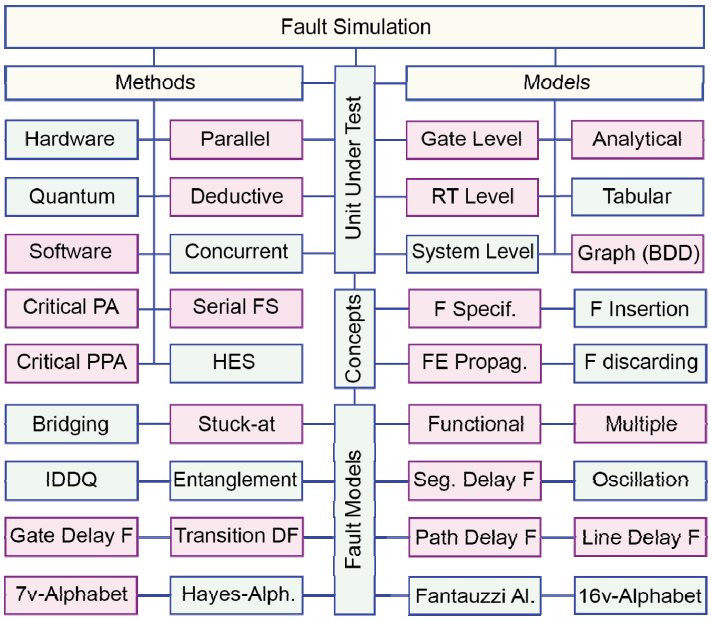


Fig. 1. Classification of fault models of digital systems [8]

The solution of the first problem is based on the following axioms:

all data is in memory;

there is no logic that cannot be implemented in memory;

there is no data that cannot be used as addresses to be processed in the memory where the logic resides;

there is no logic or functionality that cannot be implemented with a read-write transaction on memory;

the most technologically advanced memory data structure is a vector or matrix, available for fast read-write transactions.

Solving the second problem is based on the following hypotheses:

quantum computing should be based on memory without quantum logic and qubits;

memory must be stable subatomic particles;

computational processes should be based on quantum transactions between memory elements;

the source of energy for such a computer is daylight;

the speed of such a computer is determined by the light speed of quantum transactions between memory elements;

such a hypothetical computer is the meeting point of quantum (seeking determinism) and classical (seeking light speed) computing in the future.

It is necessary to resolve the contradiction between the complexity of computing devices and the lack of simple, efficient algorithms and data structures for solving problems of their verification to improve the quality of digital projects and reduce testing time through developing a vector-deductive fault simulation method based on primitive read-write transactions for analyzing logic circuits.

This requires the following:

• Development of a method for synthesizing a matrix of deductive vectors for propagating input fault vectors to the element output.

• Development of a sequencer structure for simulating faults in digital circuits based on a primitive read-write transaction of matrix memory, where combinations of faults are addresses.

• Verification and testing of models and methods for synthesizing matrices of deductive vectors and a sequencer for deductive analysis of faults in digital elements and circuits.

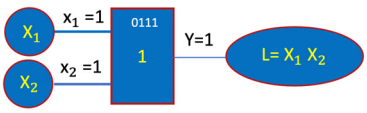
Vector method for the synthesis of deductive formulas for fault simulation. The induction is the logical conclusion from the particular to the general, where the correctness is guaranteed by a sufficient or exhaustive amount of real data. It is the basis of artificial intelligence. The deduction is the logical conclusion from the general to the, where the correctness is guaranteed by the truth of the initial axioms providing the truth of the consequences-theorems. It is the basis of deterministic computing. The mathematical basis of deductive fault simulation consists in propagating binary combinations of input faults to the output on a given input set according to the formula $L = T \oplus F$.

The essence of deductive simulation is to change the logic of the element F depending on the input conditions T. Deductive modeling, proposed exactly 50 years ago by Armstrong [26], is still the most elegant and effective tool for analyzing the quality of tests and synthesizing tables for detecting faults, tracing the propagation paths of the fault. Further, its implementation is proposed based on the vector form of logic description [7—13], which excludes logical analytical forms. This makes it possible to significantly simplify the algorithms for the synthesis of deductive models and their application for interpretative simulation of digital elements and large-scale circuits. The xor-relations triangle is the basis of the vector modification of the deductive fault simulation method. A new method for synthesizing deductive formulas is based on a vector definition of or-element (in this case) and presented in the following table (Fig. 2, *a*) [7].

Table coordinates are binary vectors, which processed according to the following rules for processing their coordinates:

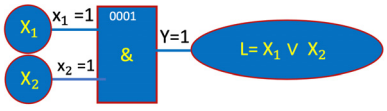
$$\vec{x}_1 = \hat{x}_2 \oplus x_i,\ x_i = \{0, 1\};$$

$$\vec{x}_1 = \hat{x}_1 \oplus x_1 = 0,\ \vec{x}_1 = \hat{x}_1 \oplus 0 = 0011 \oplus 0 = 0011;$$



Q=0111 Or				
x_1x_2	00	01	10	11
$\hat{x}_1$	0011	0011	0011	0011
$\hat{x}_2$	0101	0101	0101	0101
Q	0111	0111	0111	0111
$\vec{x}_1=\hat{x}_1\oplus x_1$	0011	0011	1100	1100
$\vec{x}_2=\hat{x}_2\oplus x_2$	0101	1010	0101	1010
$\vec{L}=Q\oplus Y$	0111	1000	1000	1000
L	$X_1 \vee X_2$	$\bar{X}_1X_2$	$X_1\bar{X}_2$	X_1X_2

a



Q=0001 and				
x_1x_2	00	01	10	11
$\hat{x}_1$	0011	0011	0011	0011
$\hat{x}_2$	0101	0101	0101	0101
Q	0001	0001	0001	0001
$\vec{x}_1=\hat{x}_1\oplus x_1$	0011	0011	1100	1100
$\vec{x}_2=\hat{x}_2\oplus x_2$	0101	1010	0101	1010
$\vec{L}=Q\oplus Y$	0001	0001	0001	0111
L	X_1X_2	$X_1\bar{X}_2$	$\bar{X}_1X_2$	$X_1 \vee X_2$

b

Fig. 2. Deductive-vector or-logic modeling (*a*), and-logic modeling (*b*) [7]

$$\vec{x}_1 = \hat{x}_1 \oplus x_1 = 1,\ \vec{x}_1 = \hat{x}_1 \oplus 1 = 0011 \oplus 1 = 1100;$$
$$\vec{x}_2 = \hat{x}_2 \oplus x_2 = 0,\ \vec{x}_2 = \hat{x}_2 \oplus 0 = 0011 \oplus 0 = 0011;$$
$$\vec{x}_2 = \hat{x}_2 \oplus x_2 = 1,\ \vec{x}_2 = \hat{x}_2 \oplus 1 = 0011 \oplus 1 = 0011.$$

Here, a new original single fault model is introduced, tied to the coordinates of the vectors-columns representing the vector component of the gate model: $\hat{x}_1 = 0011$, $\hat{x}_2 = 0101$, $Q = 0111$, derived from the columns of the truth Table

$\hat{x}_1$	$\hat{x}_2$	Q
0	0	0
0	1	1
1	0	1
1	1	1

In this case, the list-vector of detected faults is formed by performing a xor-operation between the functionality and the test set, which are also specified in vector form, for example: L = T $\oplus$ F = 01010101 $\oplus$ 10001111 = =10101010. In the coordinates of the deductive table, the procedures for executing the classical deductive algorithm for propagating faults to the output of a logic element are presented, when describing functionalities and faults in vector form. The functions of variables x_1, x_2 are set in vector form $\hat{x}_1 = 0011$, $\hat{x}_2 = 0101$. Then the inversion of the coordinates of the vectors is performed if the state of the input variable x_i is equal to one. The same applies to the out-

put variable vector Q: $L = Q \oplus Y$ when $Y = f(x_1, x_2)$. Then, using the unit constituents of the given vector L the terms of the disjunctive normal form L are written with respect to the variable vectors $\hat{x}_1$ and $\hat{x}_1$. If $L = X_1 \vee X_2$, this means that the output fault list of the gate will contain the union of the input fault lists.

The list-vector L of faults propagated to the output is written by the disjunctive normal form (DNF) of the input lists of faults presented in the last row of the table. The synthesis of deductive formulas based on vector operations to obtain a DNF for the logical element and, to transport vectors or lists of input faults, is shown in Fig. 2, *b* [7].

To compare the computational complexity of obtaining deductive formulas, the analytical classical procedure for analyzing the or-element is given below. It operates by Boolean equations, for which complex solvers are required [9, 12, 13]:

$$\mathrm{L}\,[\mathrm{T} = (00,01,10,11)\ \mathrm{F} = (\mathrm{X}_1 \vee \mathrm{X}_2)] =$$

$$= \mathrm{L}\,\{(\overline{\mathrm{x}}_1\overline{\mathrm{x}}_2 \vee \overline{\mathrm{x}}_1\mathrm{x}_2 \vee \mathrm{x}_1\overline{\mathrm{x}}_2 \vee \mathrm{x}_1\mathrm{x}_2) \wedge [(\mathrm{X}_1 \oplus \mathrm{T}_{t1} \vee \mathrm{X}_2 \oplus \mathrm{T}_{t2}) \oplus \mathrm{T}_{t3})]\} =$$

$$= (\overline{\mathrm{x}}_1\overline{\mathrm{x}}_2)\{[(\mathrm{X}_1 \oplus 0) \vee (\mathrm{X}_2 \oplus 0)] \oplus 0\} \vee (\overline{\mathrm{x}}_1\mathrm{x}_2)\{[(\mathrm{X}_1 \oplus 0) \vee (\mathrm{X}_2 \oplus 1)] \oplus 1\} \vee$$

$$\vee(\mathrm{x}_1\overline{\mathrm{x}}_2)\{[(\mathrm{X}_1 \oplus 1) \vee (\mathrm{X}_2 \oplus 0)] \oplus 1\} \vee (\mathrm{x}_1\mathrm{x}_2)\{[(\mathrm{X}_1 \oplus 1) \vee (\mathrm{X}_2 \oplus 1)] \oplus 1\} =$$

$$= (\overline{\mathrm{x}}_1\overline{\mathrm{x}}_2)(\mathrm{X}_1 \vee \mathrm{X}_2) \vee (\overline{\mathrm{x}}_1\mathrm{x}_2)(\overline{\mathrm{X}}_1 \wedge \mathrm{X}_2) \vee (\mathrm{x}_1\overline{\mathrm{x}}_2)(\mathrm{X}_1 \wedge \overline{\mathrm{X}}_2) \vee (\mathrm{x}_1\mathrm{x}_2)(\mathrm{X}_1 \wedge \mathrm{X}_2).$$

Naturally, the processing of vector models is a more technologically advanced procedure, where the computational complexity of obtaining deductive formulas for fault propagation for a function of n variables is equal to Q = = n (N + F). If we consider that N + F, as procedures for inverting vectors N and their subsequent assembly F into a vector function (union or intersection), are equal to the number of variables N + F = n, then the estimate of computational complexity in vector operations is Q = n (N+F) = n^2.

Deductive analysis of the logic circuit. A digital structure of four elements is given, each of which is represented by a Q-vector describing the states of the output variable Y in the corresponding truth table (Fig. 3, *a*). The task is to determine the list of input faults on the input binary set X = 01101, which will be propagated from each input of the circuit L = {L1, L2, L3, L4, L5} to the output L9 = F [X, L (x_i)].

To solve this problem, it is necessary to generate disjunctive normal forms for propagating input fault lists through each circuit element. The following tables (Fig. 4) contains vector procedures for processing Q-vectors of logic elements to obtain deductive formulas for each input binary set.

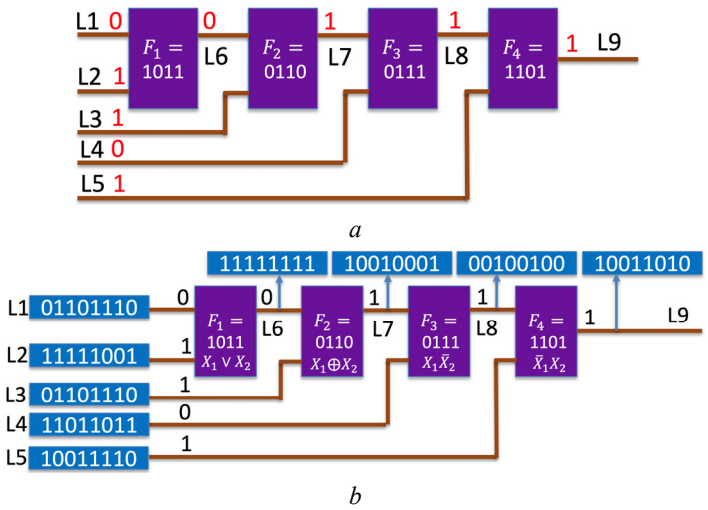


Fig. 3. Digital structure of four elements (*a*) after simulation (*b*)

Q=1011 F1				
x_1x_2	00	01	10	11
$\hat{x}_1$	0011	0011	0011	0011
$\hat{x}_2$	0101	0101	0101	0101
Q	1011	1011	1011	1011
$\vec{x}_1$=$\hat{x}_1$⊕x_1	0011	0011	1100	1100
$\vec{x}_2$=$\hat{x}_2$⊕x_2	0101	1010	0101	1010
$\vec{L}$=Q⊕Y	0100	1011	0100	0100
L	$\bar{X}_1X_2$	$X_1 \vee X_2$	$\bar{X}_1X_2$	$\bar{X}_1X_2$

Q=0110 F2				
x_1x_2	00	01	10	11
$\hat{x}_1$	0011	0011	0011	0011
$\hat{x}_2$	0101	0101	0101	0101
Q	0110	0110	0110	0110
$\vec{x}_1$=$\hat{x}_1$⊕x_1	0011	0011	1100	1100
$\vec{x}_2$=$\hat{x}_2$⊕x_2	0101	1010	0101	1010
$\vec{L}$=Q⊕Y	0110	1001	1001	0110
L	$X_1\bar{X}_2 \vee \bar{X}_1X_2$			

Q=0111 F3				
x_1x_2	00	01	10	11
$\hat{x}_1$	0011	0011	0011	0011
$\hat{x}_2$	0101	0101	0101	0101
Q	0111	0111	0111	0111
$\vec{x}_1$=$\hat{x}_1$⊕x_1	0011	0011	1100	1100
$\vec{x}_2$=$\hat{x}_2$⊕x_2	0101	1010	0101	1010
$\vec{L}$=Q⊕Y	0111	1000	1000	1000
L	$X_1 \vee X_2$	$\bar{X}_1X_2$	$X_1\bar{X}_2$	X_1X_2

Q=1101 F4				
x_1x_2	00	01	10	11
$\hat{x}_1$	0011	0011	0011	0011
$\hat{x}_2$	0101	0101	0101	0101
Q	1101	1101	1101	1101
$\vec{x}_1$=$\hat{x}_1$⊕x_1	0011	0011	1100	1100
$\vec{x}_2$=$\hat{x}_2$⊕x_2	0101	1010	0101	1010
$\vec{L}$=Q⊕Y	0010	0010	1101	0010
L	$X_1\bar{X}_2$	X_1X_2	$X_1 \vee X_2$	$\bar{X}_1X_2$

Fig. 4. Deductive-vector of main-logic modeling

To solve this problem, it is necessary to generate disjunctive normal forms for propagating input fault lists through each circuit element. The tables in the Fig. 4 contains vector procedures for processing Q-vectors of logic elements to obtain deductive formulas for each input binary set.

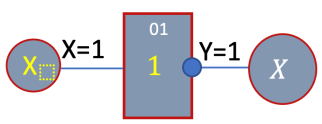
The final result of obtaining deductive formulas and applying them to propagate vectors of input fault lists to the output is shown in Fig. 3, *b*. The element F_2, whose deductive formula is represented by the second row of the table (see Fig. 4), has a unique characteristic that allows it to pass through itself only the differences (D) that are in its input lists $L = X_1\overline{X}_2 \vee \overline{X}_1X_2$. In this case, the element behaves equally invariantly to any binary input word.

The implementation of the vector-deductive method into the digital circuit design automation system will significantly simplify the methods for assessing the quality of tests, as well as reduce the time it takes for a project to evaluate its quality due to vector parallel procedures for propagating fault lists to circuit outputs.

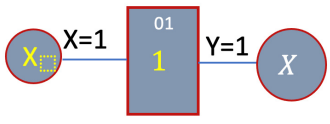
Thus, the proposed method of vector deductive analysis of logical structures differs from the well-known simulation technologies [1—5] by the high structuring of vector data, the simplest logical parallel procedures, and the high speed of obtaining deductive formulas and simulation logical elements, which makes it possible to use it for processing high-dimensional circuits. The method of vector deductive analysis of logical structures proposed in the work has no analogues in the world in terms of simplicity, performance, and its universality allows its use in various fields of human activity, including deductive analysis of logical control circuits for physical, social, virtual processes and phenomena.

Synthesis of deductive formulas of the main logical elements. Also of interest is the process of creating deductive models of the main logic elements, which can be used as a library for creating and analyzing circuits. Below are tables of vector synthesis of deductive formulas for checking logic circuits. The most primitive elements are the inverter and the repeater. Even though these are different elements, they have the same deductive formula, which allows you to carry out digital logical activity from input to output without distorting it (Fig. 5, *a*). The next frame shows the procedures for synthesizing deductive formulas for a pair of logical functions 0110 and 1001, which are mutually inverse to each other (Fig. 5, *b*).

Fig. 6, *a* explains the process of synthesizing deductive formulas for a three-input logic element given by the vector coverage 11001100. Such an element should be considered as a black box or RTL-level of function representation in relation to its structure, which can be implemented differently when specifying its behavior in a vector. Here, the result of propagating lists of activities

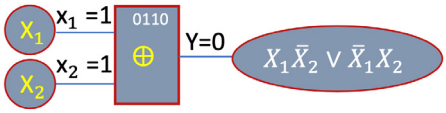


Q=01		
x	0	1
$\hat{x}_{\square}$	01	01
$\vec{x}_{\square}$	01	10
Q	01	01
$\vec{L}=Q\oplus Y$	01	10
L	X	

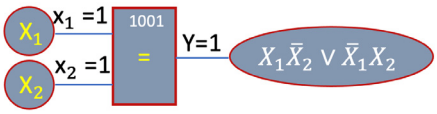


Q=10		
x	0	1
$\hat{x}_{\square}$	01	01
$\vec{x}_{\square}$	01	10
Q	10	10
$\vec{L}=Q\oplus Y$	01	10
L	X	

a



Q=0110				
x_1x_2	00	01	10	11
$\hat{x}_1$	0011	0011	0011	0011
$\hat{x}_2$	0101	0101	0101	0101
$\vec{x}_1=\hat{x}_1\oplus x_1$	0011	0011	1100	1100
$\vec{x}_2=\hat{x}_2\oplus x_2$	0101	1010	0101	1010
$L_v=\vec{x}_1\oplus\vec{x}_2$	0110	1001	1001	0110
$\vec{L}=L_v\oplus Y$	0110	0110	0110	0110
L	$X_1\bar{X}_2 \vee \bar{X}_1X_2$			



Q=1001				
x_1x_2	00	01	10	11
$\hat{x}_1$	0011	0011	0011	0011
$\hat{x}_2$	0101	0101	0101	0101
Q	1001	1001	1001	1001
$\vec{x}_1=\hat{x}_1\oplus x_1$	0011	0011	1100	1100
$\vec{x}_2=\hat{x}_2\oplus x_2$	0101	1010	0101	1010
$\vec{L}=Q\oplus Y$	0110	1001	1001	0110
L	$X_1\bar{X}_2 \vee \bar{X}_1X_2$			

b

Fig. 5. Synthesis of deductive formulas for primitive mutually inverse elements: 10 and 01 (*a*) and mutually inverse elements: 0110 and 1001 (*б*)

from the input to the output of this element is of interest. In this case, it is not interesting which paths are involved within a particular implementation of a logical element. Nevertheless, the synthesis of deductive formulas for this element showed that the formula for propagating activities has the same value $L = X_2$ for all input actions. Therefore, we can conclude that the circuit has two insignificant (redundant) inputs out of three, since they do not participate in the formation of the deductive formulas of this element.

The next frame is devoted to the process of synthesizing deductive formulas for a three-input logical RTL element given by the vector coverage 11010111 (Fig. 6, *b*). In this element, all outputs are essential, therefore, in the

Q=11001100								
$x_1x_2x_3$	000	001	010	011	100	101	110	111
$\hat{x}_1$	00001111	00001111	00001111	00001111	00001111	00001111	00001111	00001111
$\hat{x}_2$	00110011	00110011	00110011	00110011	00110011	00110011	00110011	00110011
$\hat{x}_3$	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101
Q	11001100	11001100	11001100	11001100	11001100	11001100	11001100	11001100
$\vec{x}_1=\hat{x}_1 \oplus x_1$	00001111	00001111	00001111	00001111	11110000	11110000	11110000	11110000
$\vec{x}_2=\hat{x}_2 \oplus x_2$	00110011	00110011	11001100	11001100	00110011	00110011	11001100	11001100
$\vec{x}_3=\hat{x}_3 \oplus x_3$	01010101	10101010	01010101	10101010	01010101	10101010	01010101	10101010
$\vec{L}=Q \oplus Y$	00110011	00110011	11001100	11001100	00110011	00110011	11001100	11001100
L	X_2	X_2	X_2	X_2	X_2	X_2	X_2	X_2

a

Q=11001100								
$x_1x_2x_3$	000	001	010	011	100	101	110	111
$\hat{x}_1$	00001111	00001111	00001111	00001111	00001111	00001111	00001111	00001111
$\hat{x}_2$	00110011	00110011	00110011	00110011	00110011	00110011	00110011	00110011
$\hat{x}_3$	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101
Q	11001100	11001100	11001100	11001100	11001100	11001100	11001100	11001100
$\vec{x}_1=\hat{x}_1 \oplus x_1$	00001111	00001111	00001111	00001111	11110000	11110000	11110000	11110000
$\vec{x}_2=\hat{x}_2 \oplus x_2$	00110011	00110011	11001100	11001100	00110011	00110011	11001100	11001100
$\vec{x}_3=\hat{x}_3 \oplus x_3$	01010101	10101010	01010101	10101010	01010101	10101010	01010101	10101010
$\vec{L}=Q \oplus Y$	00110011	00110011	11001100	11001100	00110011	00110011	11001100	11001100
L	X_2	X_2	X_2	X_2	X_2	X_2	X_2	X_2

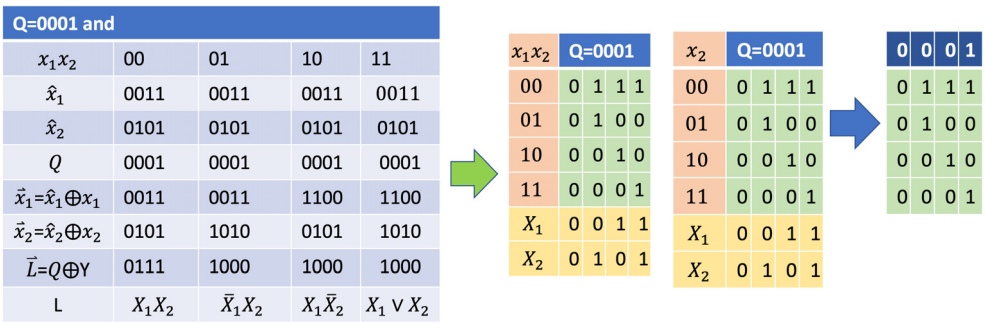
b

Fig. 6. Synthesis of deductive formulas for the three-input 11001100-element (*a*) and three-input 11010111-element (*b*)

synthesized deductive formulas, there are all input activities that are propagated to the output of this element. At the same time, it should be noted that the last row of the table indicates the already minimized deductive functions obtained from DNF using Karnaugh maps. This table is the result of executing a simple parallel algorithm on vector data structures to obtain deductive formulas for three or more variables, which can be easily coded in any programming language, including HDL — hardware description languages.

The disadvantage of this form is the use of analytical logical expressions in the form of conjunctive DFN terms for propagating fault vectors. Let's try to eliminate this drawback, leaving only deductive vectors for fault simulation.

Compactization of deductive data structures, vector-deductive sequencer. The goal is to remove all analytical expressions of deductive logic that appear in the last rows of the synthesis table. Leave only the deductive vector of fault



Q=0001 and				
x_1x_2	00	01	10	11
$\hat{x}_1$	0011	0011	0011	0011
$\hat{x}_2$	0101	0101	0101	0101
Q	0001	0001	0001	0001
$\vec{x}_1=\hat{x}_1 \oplus x_1$	0011	0011	1100	1100
$\vec{x}_2=\hat{x}_2 \oplus x_2$	0101	1010	0101	1010
$\vec{L}=Q \oplus Y$	0111	1000	1000	1000
L	X_1X_2	$\bar{X}_1X_2$	$X_1\bar{X}_2$	$X_1 \vee X_2$

a

x_1x_2	Q=0001
00	0 1 1 1
01	0 1 0 0
10	0 0 1 0
11	0 0 0 1
X_1	0 0 1 1
X_2	0 1 0 1

x_2	Q=0001
00	0 1 1 1
01	0 1 0 0
10	0 0 1 0
11	0 0 0 1
X_1	0 0 1 1
X_2	0 1 0 1

0	0	0	1
0	1	1	1
0	1	0	0
0	0	1	0
0	0	0	1

b

Fig. 7. Compactization of the matrix of deductive vectors (*a*) and its final variant (*b*)

simulation. Very often the form of the model determines the content, compactness, and speed of the analysis procedures. Therefore, for practical use, it is proposed to transform the table of deductive vector synthesis to the following form (Fig. 7, *a*, right), where the coordinates of each vector L are redefined in ascending order of input variable codes $\bar{x}_1\bar{x}_2 = 00,01,10,11$.

Here, a table convenient for the synthesis of deductive formulas is transformed into a simple matrix structure of addressable vectors for storing them in memory. A compact matrix of vectors is used to evaluate the quality of tests using the vector-deductive method of simulating single faults. How the last table works for deductive analysis is described below. The deductive vector D (x_1x_2) is found at the binary address x_1x_2, after which the projection of the unit coordinates of the D-vector onto the field of deductive variables X_1X_2 is defined to form DNF terms.

For example, x_1x_2 = 10 is the input binary variable set for the and gate. Considering it as an address, we findn the corresponding vector D (10) = 0010 in the right table. For the 1-coordinate of the D-vector, we define its projection onto the deductive variables X_1X_2 as a conjunctive term $L = X_1\bar{X}_2$. This means: to obtain the output list of element faults, it is necessary to subtract the list of faults that occur in the second input variable X_2 from the list of faults corresponding to the first variable X_1. As a rule, lists of faults for the purpose of their parallel processing are represented by vectors of unitary coding of faults of the input variable elements of a digital device.

Given that both types of variables form ordered binary addresses of deductive vectors, there is no need to store column x_1x_2 and rows X_1X_2, which can be easily generated, if necessary, as an ordered list of BCD-addresses of the corresponding rows and columns. The final version of the synthesized compact matrix of deductive vectors can be represented by the following table

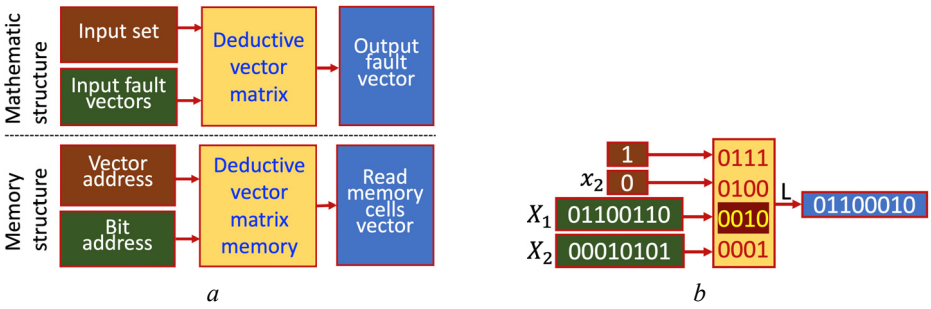


Fig. 8. Vector sequencer of deductive modeling implemented in the memory block (*a*) and an example of deductive simulation on a memory block (*b*)

(Fig. 7, *b*) with dimensions $2^n \times 2^n$, n is the number of input variables of the functional element.

The first row of the right table in Fig. 7, *b* identifies the Q-vector of the useful functionality of the element, the remaining rows are synthesized deductive vectors for simulation (propagating) faults. The matrix of deductive vectors is a kind of redundancy to a digital project, which is the cost for a fast and technological solution to the problem of assessing the quality of test patterns and building a table of fault functions to search for faults at the stage of operation of a digital product.

The proposed matrix of deductive vectors has the following properties: compactness, parallel data processing, technological placement in address memory, data uniformity in size and properties, simultaneous simulation of the correct behavior of the element and all failures of previous elements, focus on the technological solution of the problems of simulation, testing, and diagnosis of any logical systems.

The deductive simulation vector sequencer on a memory block (Fig. 8, *a*) is the simplest implementation of a computer for deductive fault simulation of digital circuit to assess the quality of tests in the class of single stuck-at faults. The main and a single memory block stores a matrix of deductive vectors of the dimension of $2^n \times 2^n$, where the first number is the number of deductive vectors for a logical element of n-variables, the second one is the dimension of each vector. Therefore, the first macro input of the memory block (Vector address) has n binary variables that allow addressing any of the 2^n deductive vectors of the matrix. The second macro input of the memory block (Byte adres) also has n, but already register binary variables, which allow addressing any of the 2^n bits of the vector, selected by the first macro input, using their binary combination. The output of the memory block (Fault list) has a bit width equal to the second macro input, determined by the power of the simu-

lated input faults that must be propagated through the element of the digital structure.

There is a one-to-one correspondence between the components of the two schemes (Mathematic structure → Memory structure): Input set → Vector address, Input fault vectors → Bit address, Deductive vector matrix → Deductive vector matrix memory, Output fault vector → Read memory cells vector. Here, the input binary set (data) entered to the gate is interpreted as the address to access the deductive vector in matrix memory. The vector of output faults propagated from element inputs is formed by reading the coordinates of a deductive vector placed in matrix memory. Unusual or paradoxical is the fact that combinations of bits of input fault vectors (data) are addresses of deductive vector bits for reading them from memory. Simply put, the faults (input data) are used as addresses to read the deductive vector bits from the matrix memory to form the output fault vector.

An example of the operation of the deductive fault simulation sequencer on the memory block is shown in Fig. 8, *b*. The first two logical inputs with their values $x_1x_2 = 10$ form the address of the deductive vector 0010. After that, each pair of input binary signals on the vectors X_1X_2 forms the bit address of the previously selected vector. The content of the selected bit at each simulation cycle forms the bit of the output fault vector obtained because of propagating the input fault lists through the element.

For example, the first signal pair $X_1X_2 = 00$ addresses 0-th memory cell of vector 0010 (in cell number format: 0123), where 0 is located, forming the first cell of the output fault vector. The second pair of signals $X_1X_2 = 10$ addresses the second memory cell of vector 0010, where 1 is located, forming the second cell of the output fault vector. Thus, pairs of signals in the same-named bits of the input fault vectors can be considered as the cell address of the selected deductive vector of the matrix. The number of simulation cycles is equal to the power of the fault vectors of the input variables. In this case, there will be eight simulation cycles, which form eight bits of the output fault vector.

Deductive-vector fault simulation of an arbitrarily complex digital circuit is reduced to a primitive memory transaction represented by an addressable read–write operation. No traditional logic is required. The same result was obtained earlier for the simulation of fault-free behavior of digital systems [3, 12, 13], where the vectors of their output states were used as a description of logical elements.

Comparison of the complexity of a functional *n*-input element F, represented by a vector of 2^n states of output coordinates and its deductive model DF, represented by MDV — a matrix of 2^n deductive vectors with a dimension of 2^n binary coordinates, is shown in Fig. 9. Here, the Bit address for the

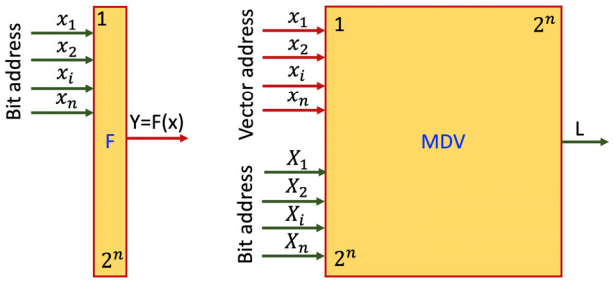


Fig. 9. Comparison of the complexity of functional and deductive elements

deductive element MDV is formed by the same-named coordinates or bits of the input fault vectors, and the Vector address is formed by the input binary set supplied to the inputs of the functional element. The bit address for the function element F is formed by the input binary word or set.

Thus, the ratio of the two models in memory is as follows: F/DF = $= 2^n / 2^n \times 2^n = 1/2^n$. The computational complexity of synthesizing a matrix of deductive vectors is estimated by the following expression: $Q = n \times 2^n + + 2^n \times 2^n$, the first term determines the complexity of generating deductive vectors unordered by addresses of input variables in register operations on vectors, the second term determines the complexity of the coordinate operations shown in Fig. 10, *a* to reorder the bits of the deductive vectors according to the order of the binary addresses constituting the input variables $\overline{x}_1\overline{x}_2$. The column order row shows the new order of the matrix columns so that all deductive vectors can be written in the order of the addresses of the input variables. In this case, there is no need to store the addresses of the input variables, we are moving from the truth table to the vector form of the description of the deductive functionality.

Algorithmically, the procedure for reordering the coordinates of deductive vectors can be represented by the following recoding table (Fig. 10, *b*). The row Reordering is actual bit addresses for binary variables $\overline{x}_1\overline{x}_2$. defines a one-to-one correspondence between the original actual positions of the deductive vector and indicates the address in the third line, to which the current bit of the vector from the first row should go.

The computational complexity of the analysis of the matrix of deductive vectors when performing deductive simulation is Q = $k \times R$, where k is the dimension of the input fault vectors, R is the duration of the operation of reading the contents of the addressable bit of the deductive vector from the MDV memory. The significant deductive redundancy of the project pays off with the quality and reliability of the digital system, which also allows generating tests,

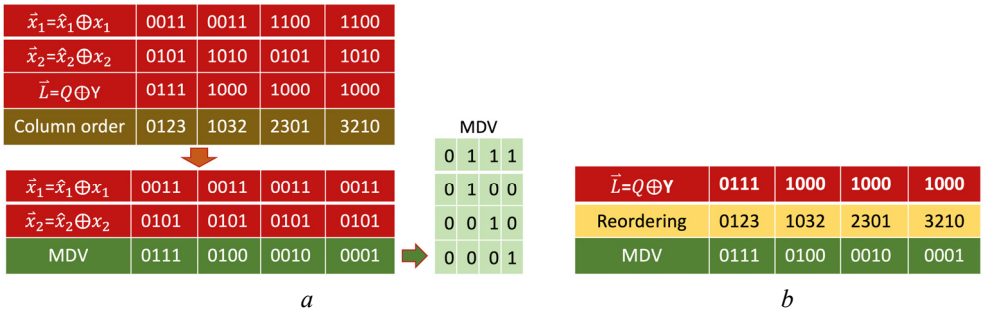


Fig. 10. Ordering (*a*) and recoding (*b*) of deductive vectors

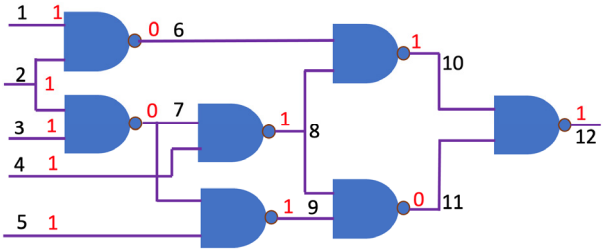
evaluating their quality, and solving any problems related to fault diagnosis in the design and operation of a digital device in critical areas of human activity.

Verification of the vector-deductive simulation method. The circuit C17 with reconverging fanouts (2, 7, 8) and an example of its vector-deductive simulation on one input set 11111 are represented in Fig. 11, *a*. In this case, a class of single stuck-at faults is considered, which are tied to input, internal or output lines (of the same potential) of the circuit.

Any line fault that is verified by this test is flagged as one in the coordinate of the fault simulation table. The real fault of the line will always be negated of its fault-free state. Vectors are also used for fault-free simulation of the circuit. This circuit uses one vector Q = 1110, which is used to simulate all elements of the circuit. The procedure of vector interpretive memory-driven simulation is described in the following works [12,13].

The right side of the table in Fig. 11, *b* contains a matrix of deductive vectors for the convenience of manual simulation. The input word x_1x_2. on the circuit element (00,01,10,11) forms the address of the deductive vector, the coordinates of which are chosen by a pair of values of the input fault vectors to form the output list of detected faults. The preparation of the fault simulation table consists in setting unit values along the diagonal of the matrix. Empty matrix coordinates denote zeros. The simulation ends when all fault vectors are sequentially generated for all 12 lines of the circuit. In fact, this means processing all seven elements of the circuit. The processing of the element with output 12 for generating the output fault list is shown in more detail (Fig. 11, *c*).

Here, the logical values of the inputs (10,11) = 10 select the deductive simulation vector 0100. By successively reading the coordinates of this vector, an output fault list is formed based on the addresses formed by the corresponding coordinate pairs (11→0, 11→ 0, 01→1, 00→0, 00→0, 10→0, 01→1, 01→1, 01→1, 10→0, 01→1, 00→01) of two input fault lists. The unit at the output of the processed element marked in red in the table is the domi-

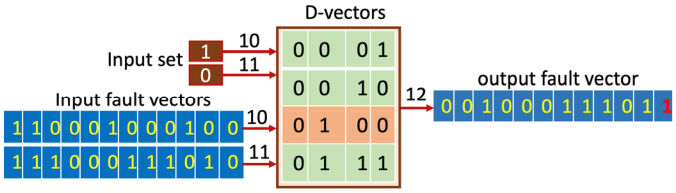


a

	1	2	3	4	5	6	7	8	9	10	11	12
States	1	1	1	1	1	0	0	1	1	1	0	1
1	1											
2		1										
3			1									
4				1								
5					1							
6	1	1				1						
7		1	1				1					
8		1	1				1	1				
9		1	1				1		1			
10	1	1				1				1		
11	1	1	1				1	1	1		1	
12			1				1	1	1		1	1
Faults			0				1	0	0		1	0

x_1x_2	Q=1110
00	0 0 0 1
01	0 0 1 0
10	0 0 0 0
11	0 0 1 1

b



c

Fig. 11. Circuit based on C17 – ISCAS-lib (*a*), its vector–deductive simulation on input set 11111 (*b*) and on input set 10 with an output 12 of the element (*c*)

nant value at the output of the simulated component. Otherwise, a fault at the output of the simulated element is always detected. Therefore, the diagonal coordinate of the matrix with the number $M_{ii}\, i = j$ does not need to be simulated as an input fault. The computational complexity of the proposed vector-deductive method is determined by the parameters of the following metric: $Q = 1/2 \times k \times n^2$, where k is the time to read a vector bit from memory, n is the number of lines in the circuit, 1/2 is half of the simulation Table.

	1	2	3	4	5	6	7	8	9	10	11	12
States	1	1	0	0	1	0	1	1	0	1	1	0
1	1											
2		1										
3			1									
4				1								
5					1							
6	1	1				1						
7		0	1				1					
8		0	0	1			0	1				
9		1	1	1			1		1			
10	1	1	0	0		1		0		1		
11	0	1	1	0			1	0	1		1	
12	1	1	1	0		1	1	0	1	1	1	1
Faults	0	0	1			1	0		1	0	0	1

x_1x_2	Q=1110
00	0 0 0 1
01	0 0 1 0
10	0 1 0 0
11	0 1 1 1

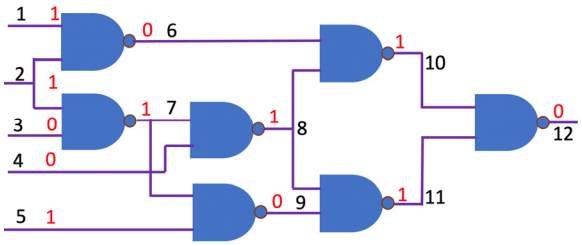


Fig. 12. Vector–deductive simulation of a circuit on an input set 11001

Another simulation example is shown in Fig. 12. The test set 110010110110 detects 9/24 = 37.5% of the faults, which are determined by negation to the state vector (fault-free state) of the circuit lines, which is needed only to decode single stuck-at faults detected on the circuit lines. Based on the results of simulation of input binary vectors, a matrix of fault coverage by test sets is generated, which has the following form for two simulated vectors:

Faults	1	2	3	4	5	6	7	8	9	10	11	12
11111			0				1	0	0		1	0
11001	0	0	1			1	0		1	0	0	1
Coverage	0	0	x			1	x	0	x	0	x	x

The symbol x = {0, 1} denotes a check on the line ≡ 0 and ≡ 1. The test is considered complete if all coordinates in the coverage line are equal to the x symbol. The method uses vector data structures, which makes it interesting from the point of view of implementation in computer memory or built-in specialized hardware calculator. The deductive simulator is free from traditional logic and uses the coordinates of the input fault vectors as bit addresses of the

deductive vectors that form the output fault list to be tested. The simulator is easily implemented in any memory, including at the quantum level, where only one read-write transaction is required.

Interpreting data as addresses can be promising when processing big data using read-write transactions on memory free of traditional logic and powerful expensive processors. The idea could create a new type of deterministic quantum computing free of qubits and quantum logic, based on stable quantum atomic memory and photonic transactions.

Conclusion

The transition to vector logic in the organization of computing and the elimination of traditional logic, presented in the form of tables and analytical expressions, made it possible to obtain the following innovative results:

1) the advantages of the vector model for a compact description of processes, phenomena, functions and structures are determined;

2) rejection of analytical expressions that require algorithmically complex calculators-analyzers, which are replaced by vector data structures for describing functional logic;

3) a vector-deductive fault simulation method based on primitive read-write transactions for analyzing logic circuits is proposed;

4) a new matrix of deductive vectors is proposed, which is characterized by the following properties: compactness, interpretability and flexibility of models saved in memory, parallel data processing based on one read-write transaction in memory, technological placement in address memory, uniformity of data in size and properties, exception phases of fault-free simulation of the element, excluding traditional logic from fault simulation procedures, fully automating the process of synthesizing the matrix of deductive vectors, focusing on the technological solving all problems of technical diagnosis;

5) the use of read-write transactions on memory in the absence of a command set focuses the new vector logic computing towards deterministic quantum architectures based on stable subatomic memory particles.

REFERENCES

1. Abramovici M., Breuer M.A. and Friedman A.D. (1998), *Digital System Testing and Testable Design,* Comp. Sc. Press.
2. Takahashi N., Ishiura N. and Yajima S. (1994), "Fault simulation for multiple faults by Boolean function manipulation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 4, pp. 531-535.
3. Dobai R., Gramatova E. (2009), "Deductive Fault Simulation for Asynchronous Sequential Circuits," *2009 12th Euromicro Conference on Digital System Design, Architectures, Methods and Tools*, pp. 459-464, doi: 10.1109/DSD.2009.129.

4. Chang H.Y., Chappell S.G, Elmendorf C.H., and Schmidt L.D. (1974), "Comparison of Parallel and Deductive Fault Simulation Methods," *IEEE Transactions on Computers*, vol. C-23, no. 11, pp. 1132-1138, Nov. 1974.
5. Pomeranz I. and Reddy S.M., "Forward-looking fault simulation for improved static compaction," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 10, pp. 1262-1265, Oct. 2001, doi: 10.1109/43.952743.
6. Navabi Z. (2011). *Digital System Test and Testable Design. Using HDL Models and Architectures*. Springer.
7. Hahanova A.V. (2022), "Developing method of vector synthesis deductive logic for computer systems fault analysis," *Herald of Advanced Information Technology*, vol. 5, no. 2, pp. 102-112, doi: 10.15276/hait.05.2022.8
8. Hahanov V., Gharibi W., Litvinova E. and Chumachenko S. (2019), "Qubit-driven Fault Simulation," *2019 IEEE Latin American Test Symposium (LATS)*, pp. 1-7, doi: 10.1109/LATW.2019.8704583.
9. Gharibi W., Devadze D., Hahanov V., Litvinova E. and Hahanov I., (2019), "Qubit Test Synthesis Processor for SoC Logic," *2019 IEEE East-West Design & Test Symposium (EWDTS)*, Batumi, Georgia, pp. 1-5, doi: 10.1109/EWDTS.2019.8884476.
10. Hahanov V. et al., (2021), "Vector-Qubit models for SoC Logic-Structure Testing and Fault Simulation," *2021 IEEE 16th International Conference on the Experience of Designing and Application of CAD Systems (CADSM)*, pp. 24-28, doi: 10.1109/CADSM52681.2021.9385266.
11. Karavay M., Hahanov V., Litvinova E., Khakhanova H. and Hahanova I. (2019), "Qubit Fault Detection in SoC Logic," *2019 IEEE East-West Design & Test Symposium (EWDTS)*, Batumi, Georgia, pp. 1-7, doi: 10.1109/EWDTS.2019.8884475.
12. Hahanov V. (2018), *Cyber Physical Computing for IoT-driven Services*, New York: Springer.
13. Hahanov V., Litvinova E., Shevchenko O., Chumachenko S., Khakhanova H. and Hahanov I. (2022), "Vector Models for Modeling Logic Based on XOR-Relations," *2022 IEEE 16th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)*, pp. 823-828, doi: 10.1109/TCSET55632.2022.9766894.
14. Biercuk M.J., Stace T.M. (2022), "Quantum Computing's Achilles. Unavoidable errors and how to fix them," *IEEE Spectrum*, pp. 28–33.
15. Versluis R., Hagen C. (2020), "Quantum computers scale up: Constructing a universal quantum computer with a large number of qubits will be hard but not impossible," *IEEE Spectrum*, vol. 57, no. 4, pp. 24-29, doi: 10.1109/MSPEC.2020.9055969.
16. Darbinyan K., Harutyunyan G., Shoukourian S., Vardanian V. and Zorian Y. (2011), "A Robust Solution for Embedded Memory Test and Repair," *2011 Asian Test Symposium*, pp. 461-462, doi: 10.1109/ATS.2011.98.
17. Harutunvan G., Vardanian V.A. and Zorian Y. (2005), "Minimal March tests for unlinked static faults in random access memories," *23rd IEEE VLSI Test Symposium (VTS'05)*, pp. 53-59, doi: 10.1109/VTS.2005.56.
18. Psarakis M., Gizopoulos D., Paschalis A. and Zorian Y., (2000),"Sequential fault modeling and test pattern generation for CMOS iterative logic arrays," *IEEE Transactions on Computers*, vol. 49, no. 10, pp. 1083-1099, doi: 10.1109/12.888044.
19. Renovell M., Portal J.M., Figueras J. and Zorian Y. (1998), "RAM-based FPGAs: a test approach for the logic," *Proceedings Design, Automation and Test in Europe*, pp. 82-88, doi: 10.1109/DATE.1998.655840.

20. Reinsalu U., Raik J., Ubar R. and Ellervee P. (2011), "Fast RTL Fault Simulation Using Decision Diagrams and Bitwise Set Operations," *2011 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems*, Vancouver, BC, pp. 164-170, doi: 10.1109/DFT.2011.42.
21. Ubar R., Devadze S., Raik J. and Jutman A., (2010), "Fast Fault Simulation for Extended Class of Faults in Scan Path Circuits," *2010 Fifth IEEE International Symposium on Electronic Design, Test & Applications*, Ho Chi Minh City, pp. 14-19, doi: 10.1109/DELTA. 2010.32.
22. Reinsalu U., Raik J. and Ubar R. (2010), "Register-transfer level deductive fault simulation using decision diagrams," *2010 12th Biennial Baltic Electronics Conference*, pp. 193-196, doi: 10.1109/BEC.2010.5631842.
23. Pomeranz I. and Reddy S.M. (2008), "Unspecified Transition Faults: A Transition Fault Model for At-Speed Fault Simulation and Test Generation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 1, pp. 137-146, doi: 10.1109/TCAD.2007.907000.
24. Pomeranz I. and Reddy S.M., "Test data compression based on output dependence," *2003 Design, Automation and Test in Europe Conference and Exhibition*, pp. 1186-1187, 2003, doi: 10.1109/DATE.2003.1253793.
25. Joe J., Mukherjee N., Pomeranz I. and Rajski J. (2022), "Fast Test Generation for Structurally Similar Circuits," *2022 IEEE 40th VLSI Test Symposium (VTS)*, pp. 1-7, doi: 10.1109/ VTS52500.2021.9794232.
26. Armstrong D.B. (1972), "A Deductive Method for Simulating Faults in Logic Circuits," *IEEE Transactions on Computers*, vol. C-21, no. 5, pp. 464-471, doi: 10.1109/T-C.1972. 223542.

Received 04.11.2022;
after revision 29.12.2022

GHARIBI Wajeb, received his PhD from the Institute of Mathematics of the National Academy of Sciences of Belarus in 1990, and is currently a professor in the Department of Computer Science at the University of Missouri in Kansas City, USA. His research interests include mathematical cybernetics, cybersecurity, machine learning, quantum computing and optimization.

HAHANOVA Anna, Cand. Sc., associate professor, graduated from Kharkiv National University of Radio Electronics (KNURE) in 2003. In 2009 she was awarded the degree of Candidate of Technical Sciences. Currently, she is an associate professor at the Design Automation Department of KNURE. Research interests are quantum and cyber-social computing, design and test of digital systems.

HAHANOV Vladimir, DSc, professor. He graduated from the Kharkov Institute of Radio Electronics in 1978. The degree of Doctor of Technical Sciences was awarded in 1996. Currently, he is a professor at the Design Automation Department of KNURE. Research interests are computer systems and networks, quantum, cyber-physical, cyber-social computing.

CHUMACHENKO Svetlana, DSc, professor. She graduated from Kharkiv State University in 1991. Doctor of Technical Sciences degree is awarded in 2008. Currently, she is the Head of the Design Automation Department of KNURE. Research interests are discrete mathematics, quantum, cyber-physical, cyber-social computing, new technologies in education.

LITVINOVA Eugenia, DSc, professor. She graduated from the Kharkov Institute of Radio Electronics in 1985. The degree of Doctor of Technical Sciences was awarded in 2010. Currently, she is a professor at the Design Automation Department of KNURE. Research interests are quantum computing, testing and diagnosis of digital systems.

HAHANOV Ivan is PhD student of the Design Automation Department of KNURE. He graduated from Kharkiv National University of Radio Electronics in 2019. Research interests are deep learning, data analysis, big data, quantum, cyber-physical, cyber-social computing, design and test of digital systems.