DOI: https://doi.org/10.15276/hait.05.2022.8
UDC 658 + 512.011 + 681.326 + 519.713

Developing method of vector synthesis deductive logic for computer systems fault analysis

Anna V. Hahanova[1)]
ORCID: https://orcid.org/0000-0002-4528-6861; anna.hahanova@nure.ua. Scopus Author ID: 8326375900
[1)] Kharkiv National University of Radio Electronics, 14, NaukyAve. Kharkiv, 61166, Ukraine

ABSTRACT

The article is devoted to the development of models and methods for fault analysis for examinate test patterns. Deductive fault simulation of digital devices is the most advanced technology that serves the field of design and testing of modern computer systems. At the same time, fault simulation solves the problem of assessing the quality of the test in the class of single constant defects. However, the computational complexity of obtaining deductive formulas, estimated as n^3, is a rather difficult task for high-dimensional RTL-level functional circuits, so the deductive method is usually used only for digital circuits represented at the gate level. Next, we propose a vector method for synthesis deductive formulas for digital schemes represented by RTL elements. This method became possible due to the element description of any complexity in the form of output states vector for combinational device. The model of xor-relationships between the wonderful logical functions (or, xor, and) of digital objects is improved, which is convoluted into zero-space. It makes possible to solve the problems of design and test, machine learning, search for similarities-differences, and destructive components in processes and phenomena. The advantages of the vector model for a compact description of objects, functions and structures are determined. It is proposed to replace analytical expressions that require algorithmically complex calculating, with vector data structures for describing functional logic. Vector-deductive method for synthesis formulas for transporting input fault lists is proposed. It has a quadratic computational complexity of register operations. The coordinate-vector model of defects is considered, not tied to input variables, which can be used for efficient processing of complex logic circuits when assessing the quality of synthesized tests. An algorithm for the synthesis of deductive vectors is presented, which differs from the known ones in the technological parallel processing simplicity of truth tables and makes it possible to create structural and logical conditions for simulating faults in digital projects of the gate, register and system description levels. An efficient method for the synthesis of a deductive truth table according to the rule $L=T\oplus F$ is proposed. It differs from the known ones by using vector-coordinate parallel xor-operation. It provides the transportation of faults through a functional element of arbitrary complexity.

Keywords: Vector form of logic; deductive matrix; truth table; deduction; deductive-vector method; digital circuit; vector model of defects

***For citation*: Hahanova A. V. Developing method of vector synthesis deductive logic for computer systems fault analysis. *Herald of Advanced Information Technology.* 2022; Vol. 5 No.2: 102–112. DOI: https://doi.org/10.15276/hait.05.2022.8**

INTRODUCTION. PROBLEM STATEMENT

Deductive fault simulation of digital devices is the most advanced technology that serves the field of design and testing of modern computer systems. At the same time, fault modeling solves the problem of assessing the quality of the test in the class of single constant defects. However, the computational complexity of obtaining deductive formulas, estimated as n3, is a rather difficult task for high-dimensional RTL-level functional circuits, so the deductive method is usually used only for digital circuits represented at the gate level.

Next, we propose a method for synthesizing deductive formulas for high-dimensional circuits represented by RTL elements. This method became possible due to the description of elements of any complexity in the form of a vector of output states of a combinational device.

LITERATURE OVERVIEW

In design and test, three main forms of describing processes and phenomena are used: tabular, analytical, graph [1, 2], [3, 4], [5, 6], [7, 8]. In this case, the matrix (table) and the vector are two forms of describing models that pass into each other. The vector is a compact form of the truth table in the form of an ordered sequence of output states, if the input address components are sorted in ascending order [1, 3], [5, 7], [8]. 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. Next, the vector appears as a compact and technological form of describing objects, functions, and structures for memory-driven computing. We are talking about the parallel processing for a finite set of logical elements represented by vectors that are in memory. Each vector defines the state of a logical element at the address of a location in memory. Therefore, such simulation is 2-3 orders of magnitude faster than existing analogues, which is important for reducing the verification time of digital devices.

Deductive modeling, proposed more than 50 years ago by Armstrong [9], and improved by many authors, including [2, 4], [10, 11], is still the most elegant and effective tool for analyzing the quality of tests and synthesizing tables for finding defects.

In [12, 13], it is proposed to improve the design process by introducing three levels of the design hierarchy and redundancy. This reduces the time-to-market design time and improves the quality of the yield project.

A technique is proposed to improve the quality of digital projects based on the technology of self-checking and self-testing circuits by introducing redundant elements [14, 15], [17, 18], [20, 21]. This idea of redundancy was used in this investigation.

The papers [16, 19], [22] develop the topic of the quality of digital projects based on redundancy, which is no more than 25 % of the functionality that solves the problems of self-testing for digital and relay-contact circuits, as well as the stable operation of circuits in the event of faults.

Further, the implementation of deductive modeling based on the vector form of the description of redundancy logic of faults transportation [5, 6], [7, 8] is proposed, which makes it possible to significantly simplify the modeling algorithms to process large-scale digital circuits.

THE AIM AND OBJECTIVES OF THE RESEARCH

Purpose of the research – developing method for synthesizing deductive formulas for high-dimensional circuits represented by RTL-elements based on the proposed model for describing elements of any complexity in the form of an output states vector for combinational device.

Objectives:

1) to improve a model of xor-relationships between wonderful Boolean functions that collapses into zero-space;

2) to develop a vector-deductive method for synthesizing formulas for transporting input fault lists. Applying a coordinate vector model of defects, not tied to input variables, for efficient processing of complex logic circuits;

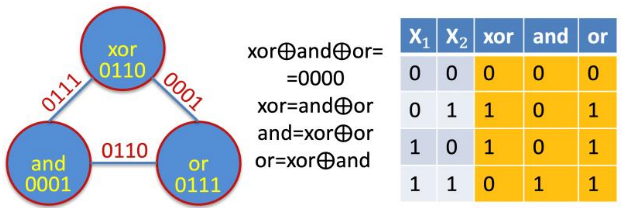
3) to develop an algorithm for the synthesis of deductive vectors. Developing an efficient method for synthesizing a deductive truth table according to the rule: L=T⨁F for transporting faults through a functional element of arbitrary complexity.

PRESENTATION OF THE MAIN RESEARCH MATERIAL

The basis for measuring processes and phenomena in a discrete (binary) space is a metric, which operates with three axioms (reflexivity, symmetry, and transitivity) of cyclic or closed interaction between 1,2,3 components. Induction (inductio) – the logical receipt of a conclusion from the particular to the general, where the correctness is guaranteed by a sufficient or exhaustive amount of factual data. Deduction (deductio) is the logical receipt of a conclusion from the general to the, where the correctness is guaranteed by the truth of the premises-axioms leading to the truth of the consequences-theorems.

The axiom of convolution (testing) of the space of processes and phenomena works for deduction: “xor-the ratio of distances between a finite number of components closed in a cycle is always equal to zero” [1]: $\oplus_{i=1}^{n} d_i = 0$.
Naturally, the xor-sum of distances between closed functional objects represented in vector form is also equal to zero.

But the fact is also true that xor is the sum of qubit [1, 8] function vectors as objects (and = 0001, or = 0111, xor = 0110), shown in Fig. 1 is zero.



X_1	X_2	xor	and	or
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	0	1	1

Fig. 1. **Convolution of boolean functions**
***Source:* compiled by the author**

Thus, it turns out that the xor-relationships of the two mentioned functions are equal to the third one. Together, three functions (Xor, And, Or) convolution to zero by means of the xor-relation: X⨁A⨁O=0.

The proposed method based on a Lemma of the convolution of three functions in two variables: there are only 4 pairs of logical functions connected by an xor-relation, which result in an xor-function:

1	2	3	4
$\oplus$=0110	$\oplus$=0110	$\oplus$=0110	$\oplus$=0110
$\wedge$=0001	$\overline{\vee}$=1000	X_1=0011	$\overline{X}_1$=1100
$\vee$=0111	$\overline{\wedge}$=1110	X_2=0101	$\overline{X}_2$=1010

Here, the first two options can indeed be considered as pairs of logical functions, the interaction of which gives the xor-function. The other two are degenerate functions of one variable that create an xor operation. If we consider that the second option is the inverse of the first one, then we can conclude that only two logical functions (and, or) are able to create an xor-vector during xor-interaction.

The practical significance of the lemma on the convolution of functions is as follows:

1) The xor operator is a unique and universal measure of the similarity-difference of any cyber-physical processes and phenomena.

2) Only two logical functions (and, or) create an xor-operation by their difference. Even more interesting is that the difference of two logical functions (and, or) is equal to their difference. Performing an xor operation on logical functions (and, or) is equal to an xor operation.

3) Two functions from one variable (inversion and repetition) also create an xor interaction.

4) The triad of logical functions (and, or, xor) forms a vector metric for calculating structural:

$$S(a,b) = a_i \underset{i=1,n}{\wedge} b_i,\ D(a,b) = a_i \underset{i=1,n}{\oplus} b_i,$$

and normalized similarity-difference estimates between processes and phenomena:

$$S_n(a,b) = \frac{\Sigma_{i=1}^{n}\left(a_i \underset{i=1,n}{\wedge} b_i\right)}{\Sigma_{i=1}^{n}\left(a_i \underset{i=1,n}{\vee} b_i\right)},\ D_n(a,b) = \frac{\Sigma_{i=1}^{n}\left(a_i \underset{i=1,n}{\oplus} b_i\right)}{\Sigma_{i=1}^{n}\left(a_i \underset{i=1,n}{\vee} b_i\right)}.$$

5) The disjunction operation (or) creates a common metric of unit values of coordinates to measure the norm of similarity-difference of two processes or phenomena.

A deductive-vector method for fault modeling is proposed.

The simulation consists of two phases:

1) Synthesis of deductive formulas based on vector description RTL-logic.

2) Simulation of logic faults when applying test actions by using the constructed deductive formulas. The formula itself hereinafter is considered as a derivative of the vector form of the functional element, fault lists input variables and input test action.

To form vectors corresponding to functions of input variables, it is necessary to synthesize or have a truth table for functions of n variables.

On Fig. 2 presented tables for four, three, two and one variables, as well as some column functions for which it is necessary to synthesize deductive vectors for modeling or transporting faults through a given functional element.

When synthesizing deductive vectors (formulas), all column vectors corresponding to input variables X_1, X_2, X_3, X_4 are used, as well as a column vector that defines a logical function, for example, A (and) or O (or), or X (xor), or F, E (other functions).

As an example, consider functions-columns-variables

$$X_1 = 0011, X_2 = 0101,$$

as well as a logical output state vector O=0111 for constructing a deductive vector.

Next, we consider the triangle of xor-relationships for the vector modification of the deductive fault modeling method using deductive formulas, the synthesis of which is proposed in Fig. 3.

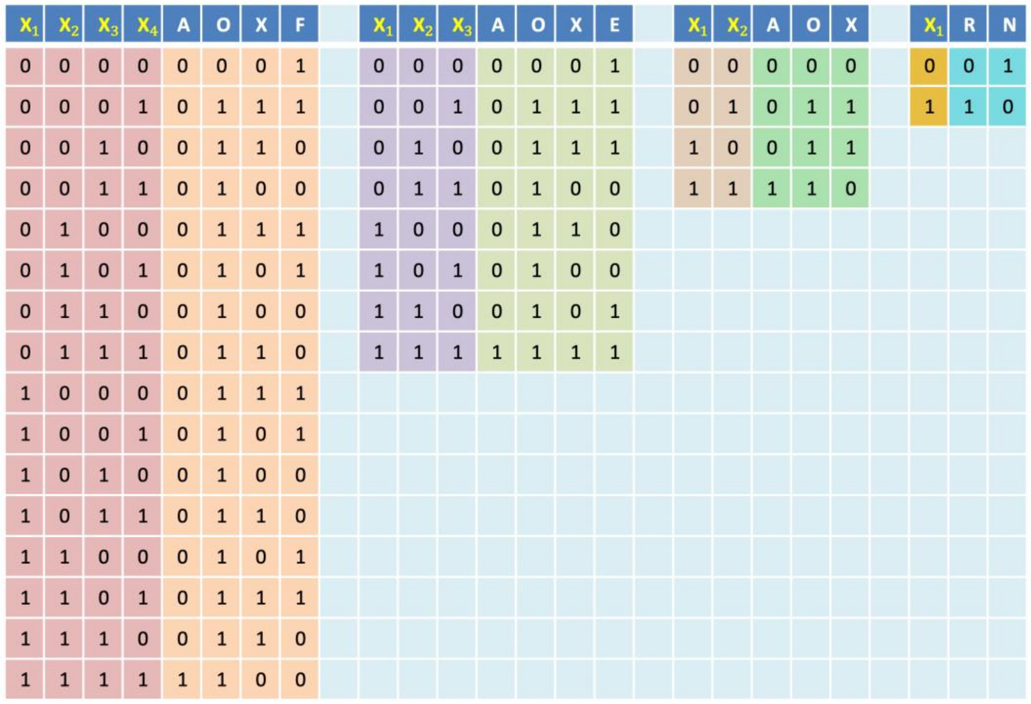
Table coordinates - binary vectors are processed according to the following rules:

$$\overrightarrow{x_1} = \hat{x}_1 \oplus x_1, x_1 = \{0,1\} \to x_1 = 1, \overrightarrow{x_1} = 0011,$$
$$\overrightarrow{x_1} = \hat{x}_1 \oplus x_1 = 1100, x_1 = 0,$$
$$\overrightarrow{x_1} = \hat{x}_1 \oplus x_1 = 0011, \overrightarrow{x_2} = \hat{x}_1 \oplus x_2 = 0,$$
$$\overrightarrow{X_2} = \hat{x}_1 \oplus 0 = 0011 \oplus 0 = 0011,$$
$$\overrightarrow{x_i} = \hat{x}_1 \oplus x_i, x_i = \{0,1\}.$$

Here, a new original single fault model is introduced, tied to the coordinates of the column vectors representing the component vectors of the gate model $x_1 = 0011, x_2 = 0101, Y = 0111$.

Derived from the columns of the following truth table:

x_1	x_2	Y
0	0	0
0	1	1
1	0	1
1	1	1



X_1	X_2	X_3	X_4	A	O	X	F
0	0	0	0	0	0	0	1
0	0	0	1	0	1	1	1
0	0	1	0	0	1	1	0
0	0	1	1	0	1	0	0
0	1	0	0	0	1	1	1
0	1	0	1	0	1	0	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	1	0
1	0	0	0	0	1	1	1
1	0	0	1	0	1	0	1
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	1	0	1
1	1	0	1	0	1	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	1	0	0

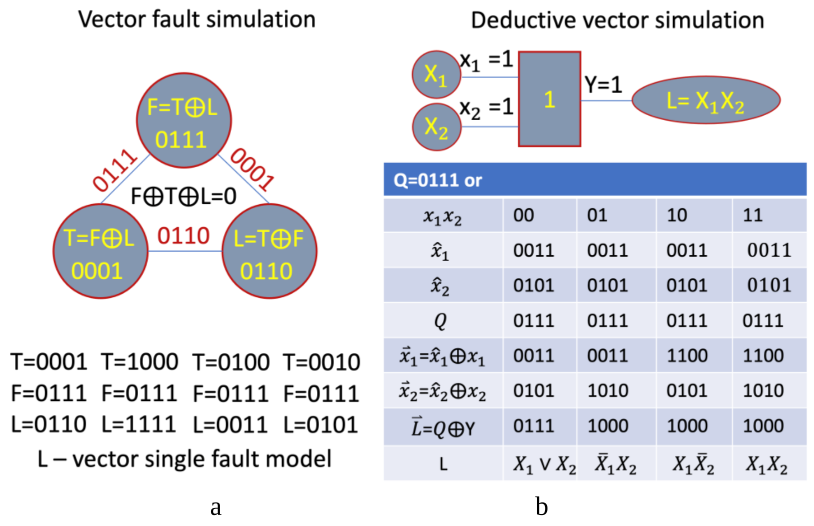
X_1	X_2	X_3	A	O	X	E
0	0	0	0	0	0	1
0	0	1	0	1	1	1
0	1	0	0	1	1	1
0	1	1	0	1	0	0
1	0	0	0	1	1	0
1	0	1	0	1	0	0
1	1	0	0	1	0	1
1	1	1	1	1	1	1

X_1	X_2	A	O	X
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

X_1	R	N
0	0	1
1	1	0

Fig. 2. Truth tables for some logic functions
Source: **compiled by the author**

On Fig. 3a, a new model of single faults is introduced, tied to the coordinates of a vector representing the model of a logical object. In this case, the list-vector of checked defects is formed by performing an xor-operation between the functionality and the test set, which are also specified in vector form. On Fig. 3b shows the procedures for synthesizing deductive formulas for the algorithm for transporting defects to the output of a logic element, but with the description of functionalities and defects in vector form. The functions of variables X_1, X_2 are set in vector form. Then the vector coordinates are inverted if the state of the input variable X_i is equal to 1. A disjunction (conjunction) of the obtained vectors is performed, and then the inversion procedure is applied to the resulting vector if the output state of the logic element is equal to one.


Vector fault simulation
F=T⊕L
0111
0111
0001
F⊕T⊕L=0
T=F⊕L
0001
0110
L=T⊕F
0110
T=0001 T=1000 T=0100 T=0010
F=0111 F=0111 F=0111 F=0111
L=0110 L=1111 L=0011 L=0101
L – vector single fault model
Deductive vector simulation
X1
X2
x1 =1
x2 =1
1
Y=1
L= X1X2
a
b

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

Fig. 3. **Deductive-vector or-logic modeling:**
a – model of single faults; b – procedures for the synthesis of deductive formulas
Source: **compiled by the author**

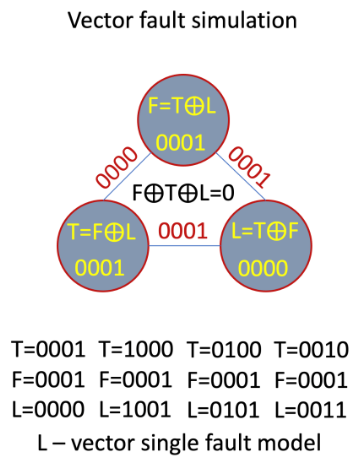
SYNTHESIS

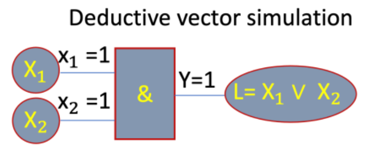
Process of synthesis for deductive fault transport functions for the logical element and is shown in Fig. 4.

Process of synthesis for deductive vectors (formulas) for transporting input fault lists through the logical element xor is shown in Fig. 5a. An interesting fact is that the analytical formula $L=\bar{X}_1X_2 \vee X_1\bar{X}_2$ for transporting lists of input defects to the output of the logical xor-element is the same for all test inputs. On Fig. 5b shows the procedure for synthesizing fault vectors L, which are dependent on the test vector T and the vector function F.

Process of synthesis for deductive functions for the repeater and inverter logic is shown in Fig. 6, which shows the same expressions for transporting fault lists to the outputs of elements without any filtering.

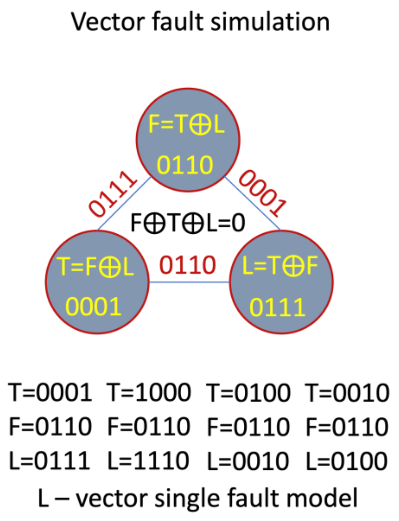
Here the lines $\vec{x}_1=\hat{x}_1\oplus x_1$, $\vec{x}_2=\hat{x}_2\oplus x_2$ form the ratio of the inversion of the state vector to the unit value of the input signal, and the line $\vec{L}=Q\oplus$Y inverts the vector if the element's output value is equal to one.

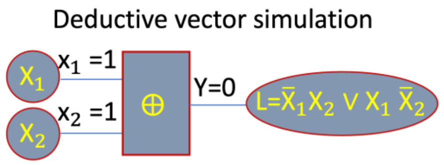

Vector fault simulation
F=T⊕L
0001
0000
0001
F⊕T⊕L=0
T=F⊕L
0001
0001
L=T⊕F
0000
T=0001 T=1000 T=0100 T=0010
F=0001 F=0001 F=0001 F=0001
L=0000 L=1001 L=0101 L=0011
L – vector single fault model



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	$\bar{X}_1X_2$	$X_1\bar{X}_2$	$X_1 \vee X_2$

Fig. 4. **Deductive-vector and-logic modeling**
Source: **compiled by the author**

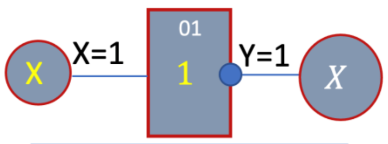

Vector fault simulation
F=T⊕L
0110
0111
0001
F⊕T⊕L=0
T=F⊕L
0001
0110
L=T⊕F
0111
T=0001 T=1000 T=0100 T=0010
F=0110 F=0110 F=0110 F=0110
L=0111 L=1110 L=0010 L=0100
L – vector single fault model



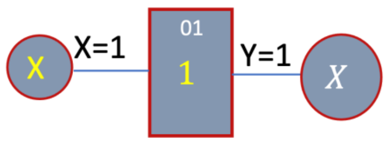
Inputs	x=00	x=01	x=10	x=11
x_1x_2	0011	0011	0011	0011
$\hat{x}_1$	0011	0011	1100	1100
$\hat{x}_2$	0101	0101	0101	0101
Q	0101	1010	0101	1010
$\vec{x}_1=\hat{x}_1\oplus x_1$	0110	1001	1001	0110
$\vec{x}_2=\hat{x}_2\oplus x_2$	0110	0110	0110	0110
$\vec{L}=Q\oplus$Y	$\bar{X}_1X_2 \vee X_1\bar{X}_2$			

a b

Fig. 5. **Deductive-vector xor-logic modeling:**
a – synthesis of deductive vectors; b – synthesis of fault vectors
Source: **compiled by the author**



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



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

Fig. 6. **Synthesis of deductive functions for the repeater and inverter**
Source: **compiled by the author**

ALGORITHM

Algorithm of the deductive vector synthesis method for simulation and/or transporting faults makes it possible to process functionally complex RTL-level digital systems.

The initial information for the execution of the vector algorithm is the truth table and the input data set. The main points of the algorithm are discussed in the following table, shown in Fig. 7, which shows the vector synthesis procedures for two input sets: 10011 and 10001, marked in red.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
X_1	X_2	X_3	X_4	F	$\bar{X}_1$	X_2	X_3	$\bar{X}_4$	L	X_1	X_2	X_3	X_4	F	$\bar{X}_1$	X_2	X_3	X_4	L
0	0	0	0	1	1	0	0	1	0	0	0	0	0	1	1	0	0	0	0
0	0	0	1	0	1	0	0	0	1	0	0	0	1	0	1	0	0	1	1
0	0	1	0	0	1	0	1	1	1	0	0	1	0	0	1	0	1	0	0
0	0	1	1	0	1	0	1	0	1	0	0	1	1	0	1	0	1	1	0
0	1	0	0	0	1	1	0	1	1	0	1	0	0	0	1	1	0	0	0
0	1	0	1	1	1	1	0	0	1	0	1	0	1	1	1	1	0	1	0
0	1	1	0	0	1	1	1	1	0	0	1	1	0	0	1	1	1	0	0
0	1	1	1	0	1	1	1	0	1	0	1	1	1	0	1	1	1	1	1
1	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	1
1	0	0	1	1	0	0	0	0	0	1	0	0	1	1	0	0	0	1	0
1	0	1	0	0	0	0	1	1	1	1	0	1	0	0	0	0	1	0	0
1	0	1	1	0	0	0	1	0	1	1	0	1	1	0	0	0	1	1	0
1	1	0	0	0	0	1	0	1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	1	0	0	1	0	0	1	1	1	0	1	0	0	1	0	1	1
1	1	1	0	0	0	1	1	1	1	1	1	1	0	0	0	1	1	0	0
1	1	1	1	1	0	1	1	0	1	1	1	1	1	1	0	1	1	1	0

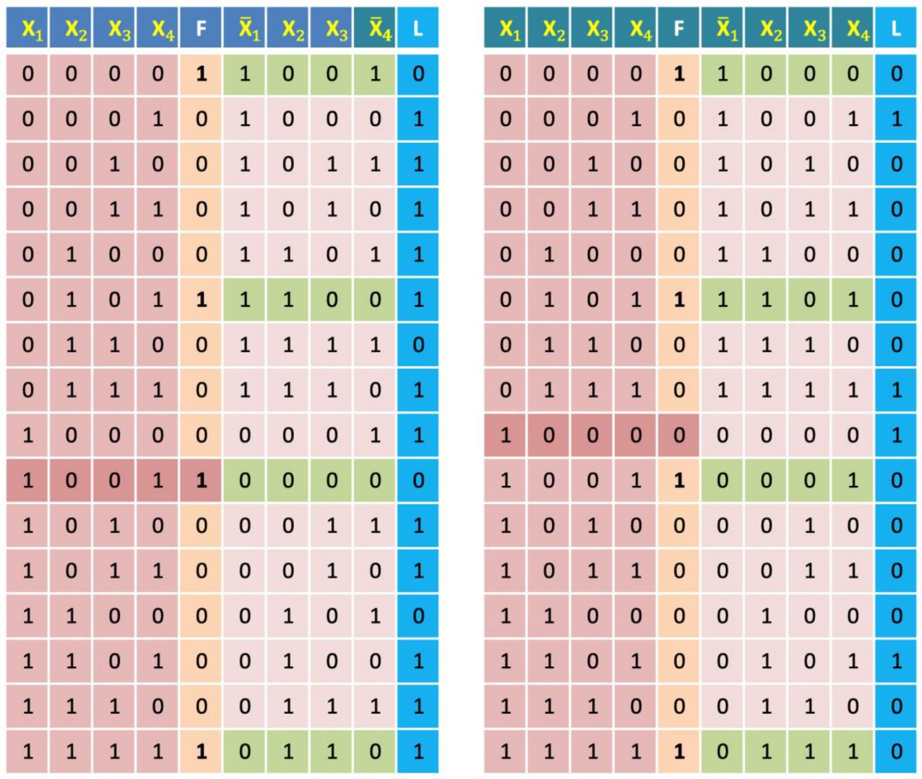


Fig. 7. **Synthesis of deductive vectors for a function of 4 variables**
Source: **compiled by the author**

Algorithm for synthesis deductive vectors using the input set 10011 as an example:

1) Setting the input vector and output state within the truth table.

2) Inverting the columns of the table in accordance with the single values of the variables of the input vector. The result of the inversion is shown in columns 6-9.

3) Determination of rows-addresses in these columns (6-9), corresponding to the unit values of the coordinates of the column vector 5 output states.

4) Formation of zero (single) coordinates, if the output state of the function is 1 (0), in the deductive vector 10 at all vector addresses marked in green, where all other coordinates are equal to one (zero).

5) Determination of the deductive DNF for a given input set (10011 and 10000) by the deductive vector, which is necessary to verify the algorithm:

$$\begin{gathered}L = [\overline{X}_1\overline{X}_2\overline{X}_3\overline{X}_4 \vee \overline{X}_1X_2\overline{X}_3X_4 \vee X_1\overline{X}_2\overline{X}_3X_4 \vee \\ \vee X_1X_2X_3X_4]\wedge(T{=}1001) = \\ = (X_1\overline{X}_2\overline{X}_3X_4 \vee X_1X_2\overline{X}_3\overline{X}_4 \vee \overline{X}_1\overline{X}_2\overline{X}_3\overline{X}_4 \vee \\ \vee \overline{X}_1X_2X_3\overline{X}_4)\oplus 1 = \\ = \overline{X}_1X_2\overline{X}_3 \vee \overline{X}_1X_4 \vee X_2X_4 \vee X_1X_3 \vee \overline{X}_2X_3 \vee \\ \vee X_3X_4 \vee X_1\overline{X}_2\overline{X}_4.\end{gathered}$$

The formation of the deductive vector for the input set 10000 is shown in columns 11-20, and the verification is presented as follows:

$$\begin{gathered}L = [\overline{X}_1\overline{X}_2\overline{X}_3\overline{X}_4 \vee \overline{X}_1X_2\overline{X}_3X_4 \vee X_1\overline{X}_2\overline{X}_3X_4 \vee \\ \vee X_1X_2X_3X_4]\wedge(T{=}1000) = \\ = (X_1\overline{X}_2\overline{X}_3\overline{X}_4 \vee X_1X_2\overline{X}_3X_4 \vee \overline{X}_1\overline{X}_2\overline{X}_3X_4 \vee \\ \vee \overline{X}_1X_2X_3X_4)\oplus 0 = \\ = X_1\overline{X}_2\overline{X}_3\overline{X}_4 \vee X_1X_2\overline{X}_3X_4 \vee \overline{X}_1\overline{X}_2\overline{X}_3X_4 \vee \\ \vee \overline{X}_1X_2X_3X_4].\end{gathered}$$

The presented algorithm for the synthesis of deductive vectors is distinguished by the technological simplicity of parallel processing of truth tables. This makes it possible to create structural and logical conditions for modeling faults in digital systems of gate, register and system description levels.

The manufacturability metric of data structures is determined by the simplicity of the algorithm. The ideal data structures are those that provide one line of code to execute the processing algorithm.

In the case of the deductive vector synthesis algorithm, for a given input set with a 1(0)-output state, only two procedures are performed in order to transport faults through a functional element:

1) Inverting the columns of the truth table by the unit coordinates of the input set – obtaining a deductive table of input values.

2) Formation of 0(1)-coordinates of the deductive vector according to the read addresses of the inverted table corresponding to 1(0)-coordinates of the vector of output states of the functionality – obtaining a vector of output values.

The algorithm can be reduced to one procedure - obtaining a deductive truth table for an input set of a given functionality.

Synthesis of the deductive truth table according to the rule:

$$L{=}T\oplus F \rightarrow L_i = T_i \oplus F_i.$$

Here T is a test data set (truth table row) by input and output coordinates; T_i is the coordinate of the input set; F a truth table for a given functionality; F_i is the vector column of the truth table; L deductive truth table; L_i is the column vector of the deductive truth table; $\oplus$ – a coordinate-vector operation on columns.

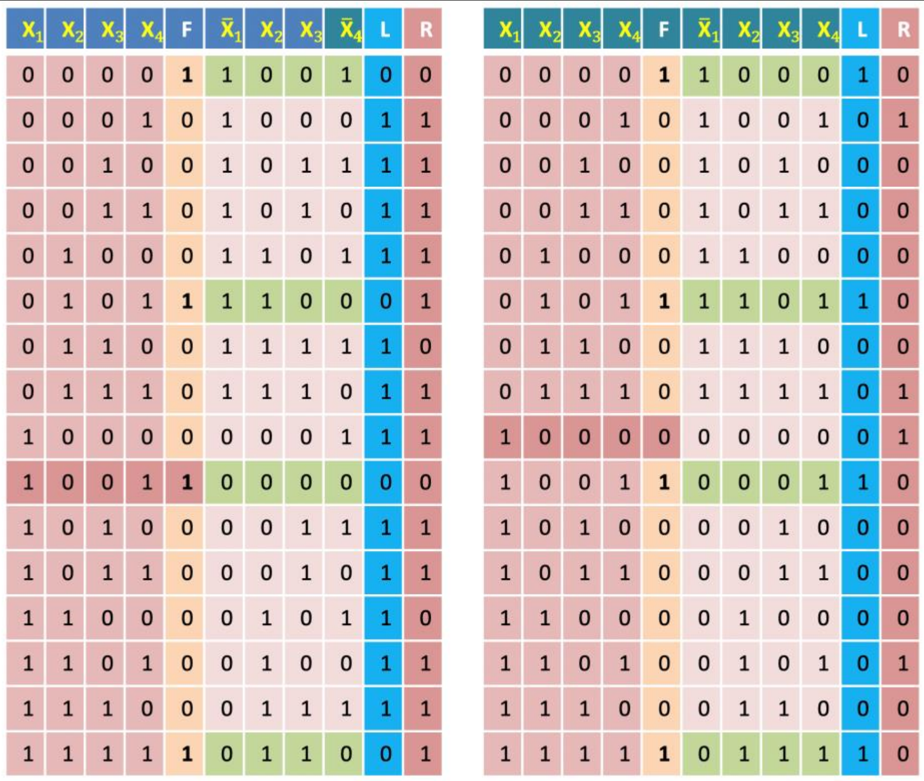
To explain the last algorithm, in one line of code, the column R is added (Fig. 8), which denotes the states of the deductive truth table relative to the order of the input sets of the original functionality F.

After receiving a deductive truth table unordered by input codes, it is restructured to the form of the original table, but with a different (deductive) output vector R=f(F). The result – the deductive table – should be considered columns 1-4, 10, as well as columns 12-15, 22. In fact, the whole algorithm is reduced to the synthesis of the deductive output state column L (R) for the initial truth table F based on the test set T.

RESULTS COMPARISION

To compare the computational complexity of obtaining deductive formulas, the following is an analytical classical procedure for analyzing the or-element, which operates with Boolean equations, for which complex solvers are required [1, 8]:

$$\begin{aligned}&L[T = (00,01,10,11), F = (X_1 \vee X_2)] = \\ &= L\{(\overline{x}_1\overline{x}_2 \vee \overline{x}_1x_2 \vee x_1\overline{x}_2 \vee x_1x_2) \wedge [(X_1 \oplus T_{t1} \vee X_2 \oplus T_{t2}) \oplus T_{t3})]\} = \\ &= (\overline{x}_1\overline{x}_2)\{[(X_1 \oplus 0) \vee (X_2 \oplus 0)] \oplus 0\} \vee (\overline{x}_1x_2)\{[(X_1 \oplus 0) \vee (X_2 \oplus 1)] \oplus 1\} \vee \\ &\vee (x_1\overline{x}_2)\{[(X_1 \oplus 1) \vee (X_2 \oplus 0)] \oplus 1\} \vee (x_1x_2)\{[(X_1 \oplus 1) \vee (X_2 \oplus 1)] \oplus 1\} = \\ &= (\overline{x}_1\overline{x}_2)(X_1 \vee X_2) \vee (\overline{x}_1x_2)(\overline{X}_1 \wedge X_2) \vee (x_1\overline{x}_2)(X_1 \wedge \overline{X}_2) \vee (x_1x_2)(X_1 \wedge X_2).\end{aligned}$$



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
X_1	X_2	X_3	X_4	F	$\bar{X}_1$	X_2	X_3	$\bar{X}_4$	L	R	X_1	X_2	X_3	X_4	F	$\bar{X}_1$	X_2	X_3	X_4	L	R
0	0	0	0	**1**	1	0	0	1	0	0	0	0	0	0	**1**	1	0	0	0	1	0
0	0	0	1	0	1	0	0	0	1	1	0	0	0	1	0	1	0	0	1	0	1
0	0	1	0	0	1	0	1	1	1	1	0	0	1	0	0	1	0	1	0	0	0
0	0	1	1	0	1	0	1	0	1	1	0	0	1	1	0	1	0	1	1	0	0
0	1	0	0	0	1	1	0	1	1	1	0	1	0	0	0	1	1	0	0	0	0
0	1	0	1	**1**	1	1	0	0	0	1	0	1	0	1	**1**	1	1	0	1	1	0
0	1	1	0	0	1	1	1	1	1	0	0	1	1	0	0	1	1	1	0	0	0
0	1	1	1	0	1	1	1	0	1	1	0	1	1	1	0	1	1	1	1	0	1
1	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	1
1	0	0	1	**1**	0	0	0	0	0	0	1	0	0	1	**1**	0	0	0	1	1	0
1	0	1	0	0	0	0	1	1	1	1	1	0	1	0	0	0	0	1	0	0	0
1	0	1	1	0	0	0	1	0	1	1	1	0	1	1	0	0	0	1	1	0	0
1	1	0	0	0	0	1	0	1	1	0	1	1	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	1	1	1	1	0	1	0	0	1	0	1	0	1
1	1	1	0	0	0	1	1	1	1	1	1	1	1	0	0	0	1	1	0	0	0
1	1	1	1	**1**	0	1	1	0	0	1	1	1	1	1	**1**	0	1	1	1	1	0

Fig. 8. **Synthesis of deductive truth table by L=T⊕F**

***Source:* compiled by the autho**

Naturally, the processing of vector models is a more technologically advanced procedure, where the computational complexity of obtaining deductive formulas for defect transportation for a function of n variables is equal to Q=n(N+F). If we take into account 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 computational complexity estimate in vector operations is Q=n(N+F)= n^2. At the same time, it should be noted that the analytical algorithm for the synthesis of deductive formulas has cubic computational complexity.

CONCLUSION

Thus, the proposed synthesis of the deductive truth table according to the L=T⊕F rule is the most efficient method, which differs from the known ones by using a single vector-coordinate parallel xor-operation, which ensures the transportation of faults through a functional element of arbitrary complexity. The full deductive matrix Q of a functional element of arbitrary complexity from n variables for the analysis 2^n of input actions to obtain the quality of the test has the dimension Q= $2^n \times (2^n$+n).

The practice of applying deductive analysis based on L=T⊕F in a broad sense is determined by the need to diagnose destructive components in quantum, digital, cyberspace, physical, technical, energy, transport, network, social, legal, economic processes, and phenomena.

The scientific novelty of the research is determined by:

1) improvement of the xor-relationship model between the logical functions or, xor, and, which collapses into zero-space;

2) development of a vector-deductive method for synthesizing formulas for transporting input fault lists, which has a quadratic computational complexity of register operations;

3) the development of an effective method for the synthesis of a deductive truth table according to the rule L=T⊕F, which differs in the use of a single vector-coordinate parallel xor-operation, which makes it possible to ensure the transportation of faults through a functional element of arbitrary complexity.

The practical significance of the research results:

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

2) to replace analytical expressions, that require algorithmically complex calculating, with vector data structures for describing functional logic;

3) using coordinate-vector model for defects that is not tied to input variables, which can be used for efficient processing of complex logic circuits when assessing the quality of generated tests;

4) development of an algorithm for the synthesis of deductive vectors, which differs from the known technological simplicity of parallel processing of truth tables. It makes it possible to create structural and logical conditions for modeling faults in digital projects of the gate, register and system description levels.

The fault analysis time is reduced due to the deductive redundancy of the element and circuit models.

The obtained results can be used for technological solution of modeling, testing and diagnostics problems.

REFERENCES

1. Hahanov, V. “Cyber physical computing for IoT-driven services”. *Publ. Springer*. New York: 2018.

2. Liu, T. Yu, T. Wang, S. & Cai, S. “An efficient degraded deductive fault simulator for small-delay defects”. *In IEEE Access.* 2020; Vol.8: 204855–204862. DOI: https://doi.org/10.1109/ACCESS.2020.3037292.

3. Hahanov, V., Chumachenko, S., Iemelianov, I., Hahanov, V., Larchenko, L. & Daniyil, T. “Deductive qubit fault simulation”. *14th International Conference: The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM).* 2017. p. 256–259. DOI: https://doi.org/10.1109/CADSM.2017.7916129.

4. Reinsalu, U., Raik, J. & Ubar, R. “Register-transfer level deductive fault simulation using decision diagrams”. *12th Biennial Baltic Electronics Conference.* 2010. p. 193–196. DOI: https://doi.org/10.1109/BEC.2010.5631842.

5. Hahanov, V., Hacimahmud, A. V., Litvinova, E., Chumachenko, S. & Hahanova, I. “Quantum deductive simulation for logic functions”. *IEEE East-West Design & Test Symposium (EWDTS*), 2018. p. 1–7. DOI: https://doi.org/10.1109/EWDTS.2018.8524619.

6. Dobai, R. &. Gramatova, E. “Deductive fault simulation for asynchronous sequential circuits“, *12th Euromicro Conference on Digital System Design, Architectures, Methods and Tools*. 2009. p. 459–464. DOI: https://doi.org/10.1109/DSD.2009.129.

7. Hahanov, V., Gharibi, W., Litvinova, E. & Chumachenko, S. "Qubit-driven fault simulation”, *IEEE Latin American Test Symposium (LATS).* 2019. p. 1–7. DOI: https://doi.org/10.1109/LATW.2019.8704583.

8. Hahanov, V., Garibi, W., Chumachenko, S., Litvinova, E., Hacimahmud, A. V., Salih, T. H. & Hahanov, I. “Vector-qubit models for SoC logic-structure testing and fault simulation“. *IEEE 16th International Conference on the Experience of Designing and Application of CAD Systems (CADSM)*. 2021. p. 24–28. DOI: https://doi.org/10.1109/CADSM52681.2021.9385266.

9. Abramovici, M., Breuer, M. A. & Friedman, A. D. “Digital system testing and testable design”. *Comp. Sc. Press*. 1998.

10. Pomeranz, I. & Reddy, S. M. “Forward-looking fault simulation for improved static compaction“, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 2001; Vol. 20 No.10: 1262–1265. DOI: https://doi.org/10.1109/43.952743.

11. Pomeranz, I. & Reddy, S. M. “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 System*s. 2008; Vol. 27, No.1: 137–146. DOI: https://doi.org/10.1109/TCAD.2007.907000.

12. Larshin, V. P., Lishchenko, N. V., Babiychuk, O. B. & Piteľ, Ján. “Computer-aided design and production information support”. *Herald of Advanced Information Technology*. *Nauka i Tekhnika*. Odessa: Ukraine. 2021; Vol. 4 No. 2: 111–122. DOI: https://doi.org/10.15276/hait.02.2021.1.

13.Drozd, O. V., Rucinski, A., Zashcholkin, K. V., Drozd, M. O. & Sulima, Y. Y. “Improving FPGA components of critical systems based on natural version redundancy”, *Applied Aspects of Information Technology*. 2021; Vol.4 No. 2: 168–177. DOI: https://doi.org/10.15276/aait.02.2021.4.

14. Antoniuk, V. V., Drozd, M. O. & Drozd, O. V. “Power-oriented checkability and monitoring of the current consumption in FPGA projects of the critical applications”. *Applied Aspects of Information Technology.* 2019; Vol.2 No.2: 105–114. DOI: https://doi.org/10.15276/aait.02.2019.2.

15. Kovalev, I. S., Drozd, O. V., Rucinski, A., Drozd, M. O., Antoniuk, V. V. & Sulima, Y. Y. “Development of computer system components in critical applications: problems, their origins and solutions”. *Herald of Advanced Information Technology*. 2020; Vol.3 No.4: 252–262. DOI: https://doi.org/10.15276/hait.04.2020.4.

16. Efanov, D. V., Sapozhnikov, V. V. & Sapozhnikov, Vl. V. “Synthesis of self-checking combination devices based on allocating special groups of outputs”. *Automation and Remote Control*. 2018; Vol. 79 Issue 9: 1609–1620. DOI: https://doi.org/10.1134/S0005117918090060.

17. Drozd, A. & Antoshchuk, S. “New on-line testing methods for approximate data processing in the computing circuits”. *6th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Prague, Czech Republic*. 2011. p. 291–294. DOI: https://doi.org/10.1109/IDAACS.2011.6072759.

18. Drozd, M. & Drozd, A. “Safety-related instrumentation and control systems and a problem of the hidden faults”. *The 10th International Conference on Digital Technologies*. *Slovak Republic*. 2014. p. 137–140. DOI: https://doi.org/10.1109/DT.2014.6868692.

19. Efanov, D. V. Sapozhnikov, V. V. & Sapozhnikov, Vl. V. “Organization of a fully self-checking structure of a combinational device based on searching for groups of symmetrically independent outputs”, *Automatic Control and Computer Sciences*, 2020; Vol. 54 Issue 4: 279–290. DOI: https://doi.org/10.3103/S0146411620040045.

20. Drozd, A., Drozd, J., Antoshchuk, S., Nikul, V. & Al-dhabi, M. “Objects and methods of on-line testing: main requirements and perspectives of development”. *Proc. IEEE East-West Design & Test Symposium*. Yerevan: Armenia. 2016. p. 1– 5. DOI: https://doi.org/10.1109/EWDTS.2016.7807750.

21. Drozd, O., Zashcholkin, K., Martynyuk, O., Ivanova, O. & Drozd, J. “Development of checkabilityin fpga components of safety-related systems. *CEUR Workshop Proceedings*. 2020; Vol. 2762: 30–42. – Available from: http://ceur-ws.org/Vol-2762/paper1.pdf. – [Accessed Dec 2020].

22. Efanov, D. V., Sapozhnikov, V. V. & Sapozhnikov, Vl. V. “Boolean-complement based fault-tolerant electronic device architectures”. *Automation and Remote Control*. 2021; Vol. 82 Issue 8: 1403–1417. DOI: https://doi.org/10.1134/S0005117921080075.

Conflicts of Interest: The authors declare no conflict of interest

Received 15.01.2021
Received after revision 12.03.2021
Accepted 16.03.2021

DOI: https://doi.org/10.15276/hait.05.2022.8
УДК 658 + 512.011 + 681.326 + 519.713

Розробка векторного синтезу дедуктивної логіки для аналізу несправностей комп’ютерних систем

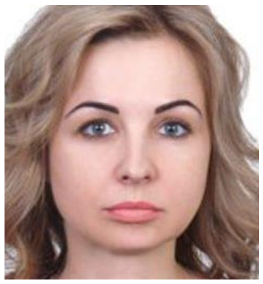
Ганна Володимирівна Хаханова
ORCID: https://orcid.org/0000-0002-4528-6861; anna.hahanova@nure.ua. Scopus Author ID: 8326375900
Kharkiv National University of Radio Electronics, 14, Nauky Ave. Kharkiv, 61166, Ukraine

АНОТАЦІЯ

Стаття присвячена розробці моделей та методів для аналізу несправностей. Дедуктивне моделювання несправностей цифрових пристроїв є передовою технологією, яка обслуговує область проектування і тестування сучасних комп'ютерних систем. При цьому моделювання несправностей розв’язує задачу оцінки якості тесту у класі одиночних константних дефектів. Однак обчислювальна складність отримання дедуктивних формул, що оцінюється як n^3, є досить складним завданням для функціональних схем RTL-рівня великої розмірності, тому дедуктивний метод, як правило, використовується тільки для цифрових схем, представлених на рівні вентилів. Далі пропонується метод синтезу дедуктивних формул для схем великої розмірності, представлених RTL-елементами. Даний метод став можливим завдяки опису елементів будь-якої складності у вигляді вектора вихідних станів комбінаційного пристрою. Удосконалюється модель xor-відношень між «чудовими» логічними функціями цифрових об'єктів, яка згортається в нуль-простір, що дає можливість розв’язувати задачі технічної діагностики, машинного навчання, пошуку подібності-відмінності, діагностування деструктивних компонентів у процесах та явищах. Визначаються переваги векторної моделі для компактного опису об'єктів, функцій та структур. Пропонується замінити аналітичні вирази, що вимагають алгоритмічно складних обчислювачів-аналізаторів, на векторні структури даних для опису функціональної логіки. Пропонується векторно-дедуктивний метод синтезу формул для транспортування списків вхідних несправностей, який має квадратичну обчислювальну складність регістрових операцій. Розглядається координатно-векторна модель дефектів, не прив'язана до вхідних змінних, яка може бути використана для ефективної обробки складних логічних схем для оцінки якості синтезованих тестів. Надається алгоритм синтезу дедуктивних векторів, який відрізняється від відомих технологічною простотою паралельної обробки таблиць істинності та дає можливість створювати структурно-логічні умови для моделювання несправностей у цифрових проектах вентильного, регістрового та системного рівнів опису. Пропонується ефективний метод синтезу дедуктивної таблиці істинності за правилом L=T⊕F, який відрізняється від відомих застосуванням єдиної векторно-координатної паралельної xor-операції, що забезпечує транспортування несправностей через функціональний елемент довільної складності.

Keywords: векторна форма логіки; дедуктивна матриця; таблиця істинності; дедукція; дедуктивно векторний метод; цифрова схема; векторна модель дефектів

ABOUT THE AUTHOR



Anna V. Hahanova – Candidate of Engineering Sciences, Associate Professor of the Design Automation Department. Kharkiv National University of Radio Electronics,14, Nauky Ave. Kharkiv, 61166, Ukraine

ORCID: https://orcid.org/0000-0002-4528-6861; anna.hahanova@nure.ua. Scopus Author ID: 8326375900.

***Research field*:** Cyber-physical; cyber-social computing; pattern recognition and machine learning.

Ганна Володимирівна Хаханова – канд. техн. наук, доцент, кафедра Автоматизації проектування обчислювальної техніки. Харківський національний університет радіоелектроніки, пр. Науки, 14, Харків, 611166, Україна