IAES International Journal of Robotics and Automation (IJRA)
Vol. 15, No. 2, June 2026, pp. 415~426
ISSN: 2722-2586, DOI: 10.11591/ijra.v15i2.pp415-426

Vector logic for robotic system on chip design and test

Vladimir Hahanov, Svetlana Chumachenko, Eugenia Litvinova,
Andrii Voronov, Oleh Demchenko, Nataliya Maksymova
Department of Design Automation, Kharkov National University of Radio Electronics, Kharkiv, Ukraine

Article Info

Article history:

Received Dec 16, 2025
Revised Feb 13, 2026
Accepted Mar 6, 2026

Keywords:

Interconnect bus
Modelling for simulation
Robots
Structures and functions
System on chip
Test map
Vector logic

ABSTRACT

Artificial Intelligence and vector logic of computing do not contradict but cooperate and enrich each other. Logic is the law of existence and development of emerging computing. Logic is functions and structures, models and algorithms, phenomena and processes. Any computing, including artificial intelligence, is logic and nothing else. Emerging computing devices today have hundreds of systems on a chip and memory blocks, which are interconnected by thousands of connecting wires. This encompasses all the logic, functionalities, and structures, which are subject to testing by system methods. To achieve this, a logic vector serves as a generic form for describing functions, structures, and buses in modeling for the simulation of test sets and logic faults as address. Chip-let Interconnect bus is also a logical functionality or structure. They must be tested to diagnose defects by system logic mechanisms. The latter involves modeling to automatically obtain data structures, followed by good-value simulation and simulation of all fault combinations, such as addresses, on the buses segment. For this purpose, vector logic is used to describe functionalities and structures, models and algorithms, faults and tests. Mechanisms and application that assume a harmonious relationship between the model and the algorithm for their processing are considered.

Corresponding Author:

Vladimir Hahanov
Department of Design Automation, Kharkov National University of Radio Electronics
Kharkiv, Ukraine
Email: vladimir.hahanov@nure.ua

1. INTRODUCTION

Robots are autonomous agents that automatically perform spatial services with high accuracy within a specified time frame [1]. The robot metric is the maximum time autonomy and spatial accuracy of the task [2]. The robot's metric is achieved through an efficient power supply and energy-saving control devices [3]. These devices today include i) system-on-chips (SoC) [4], designed according to specific architecture, ii) in-memory computing [5] utilizing read-write transactions on iii) vector logic [6]. The integration of these three components as illustrated Figure 1 enables [5], [7] the reduction of latency (11×) for generating actuator signals for the robot, as well as energy savings (3–5×) in non-processor calculations related to controlling an autonomous agent or robot [8]. An automated SoC device for robot control requires simple testing mechanisms to verify its functionality [9]. Vector logic [6] serves as an effective tool for testing the robot SoC, used both to describe the robot's functional properties within the SoC and as a mechanism for modeling, simulating, testing, and diagnosing SoC logic circuits and robot control [10], [11]. Therefore, vector logic functions not only as the core of the autonomous system's functionality but also provides efficient mechanisms (models and algorithms) for testing that system. In-memory computing reduces latency by 10× and energy consumption by 3–5× compared to processor-based computing [5]. Read-write transactions on

vector logic eliminate (GL, RTL) circuits from in-memory computing, making it even more economical (2×) in terms of latency and power consumption [5], [12].

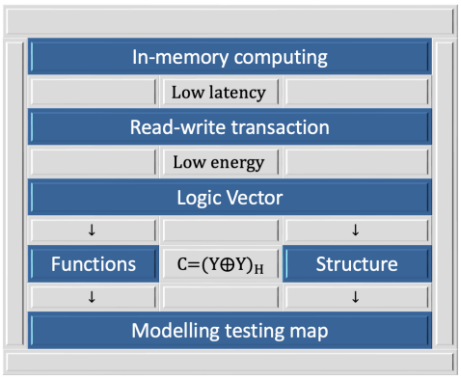


Figure 1. The map of vector logic in-memory computing

Vector logic [6] is an extensive, exponentially redundant framework for designing and testing functions, structures, models, and algorithms. Vector logic implemented in in-memory computing solves all known problems using read-write transactions without processor instructions [6]. This makes vector logic the foundation for the future of energy-efficient and mass energy computing. Metric vector logic is a truth table with explicit binary addresses that solves combinatorial problems using linear-complexity algorithms. The truth table has been known for over a century [13]. It was not used in computing because of its exponential redundancy in describing Boolean functions. Researchers aimed to minimize the truth table to find the smallest analytical conjunctive and disjunctive forms. Minimizing Boolean function descriptions was tied to the cost-effective use of slow, expensive memory [14]. Now, the scenario has changed; memory has become as inexpensive and fast as processor logic. Due to its exponential redundancy, vector logic or truth tables eliminate the need for complex algorithms to analyze data structures. Consequently, vector logic reduces the time needed to generate inferences during simulation or robot control with activation signals. Vector logic is designed to leverage memory redundancy efficiently to minimize modeling time for simulating various intelligent computing systems, including quantum and Artificial Intelligence systems [15]. A neural network is simply a structure of interconnected vertex layers represented by matrices, which can be considered a form of logical vector. The logical vector [6] is the computing genome. A logical vector is a sequence of bits of length n, where n is the number of binary variables. The metric of a logical vector is implicitly defined by binary addresses and the vector itself. A truth table is a Boolean vector with explicit binary addresses for its bits. A Boolean vector is a compact record of a truth table that lacks explicit binary addresses. Binary addresses serve as i) input effects of logical functions; ii) test tools for verifying functionality; iii) a complete set of fault conditions on individual bits of a binary address; iv) identifiers for graph vertices connected by arcs marked with unit coordinates of the logical vector; and v) a comprehensive set of difference-similarity properties used for diagnostics. A logical vector offers a parallel solution to any logical problem through a single read-write transaction within an in-memory computing system. It surpasses schematic solutions for logical problems by generating test maps for any function without simulation algorithms, relying solely on three matrix operations. Vector logic represents the simplest, most energy-efficient, and time-saving form of intelligent computing for the future. Testing the features and structures of robotic SoCs is a classic verification task [16] that can be addressed during the design phase using methods such as manually generating valid tests with industrial simulation tools [17]. The importance of testing highlights the need for new approaches to reduce SoC testing time and simplify modeling for simulation [18]. The optimal solution involves modeling a test card, function, or structure [19]–[21] without simulation algorithms, utilizing just three matrix operations [6]. This is possible because of the exponential redundancy inherent in vector logic used for describing functionalities [21]. Redundancy removes the need for fault simulation during comprehensive test development [22]. However, a challenge of vector-logical testing is the difficulty in visually representing input and output effects when modeling high-dimensional logical functions [23].

Scientific novelty involves a logical distinction that results in resource savings or the creation of a new quality. Engineering translates scientific findings into repeated practical applications. Scientific research results should be presented aligned with these criteria. Intelligent computing [15] represents a harmonious integration of hardware and software between a model and an algorithm, focused on saving time and energy

by efficiently utilizing space and matter. Artificial intelligence assists humans in making final decisions—this is today' s cave computing [24], [25] for human eyes only. Still, emerging from cyberspace are the beginnings of a new AI activation computer, known as an autonomous agent. What' s new here? No issue. Classical von Neumann computing involves monitoring register and memory states to generate control actions [24]. This idea is 80 years old. The newest concepts are often just old ideas revisited. The goal of creating an activation computer (autonomous agent) is to replace humans not only in routine tasks but also to act as strategic decision- making assistants. Ultimately, autonomous agents aim to replace humans as the unreliable control point in social, technical, scientific, educational, technological, engineering, and mathematical activities. To organize hierarchical memory and connect all units and devices, it is necessary to test and diagnose both blocks and buses. The 3-D design of the Chiplet today includes up to 100, 100,000 interconnects, enabling the operation of the computing system [26]. Currently, this issue is addressed by dividing conductors into clusters and testing each cluster separately [27]. Additionally, at the request of the IEEE, leading scientists are developing a standard, IEEE P 3405, for tire testing [28]. The problem exists, but a systemic solution has not yet been implemented. A system solution is a model of an object or process that explicitly or implicitly captures faults, possible violations, or malfunctions. Modeling such faults, if not expressly specified, should be automated using simple algorithms or not at all. This type of modeling is also considered a smart computing mechanism today. There is no need to involve Artificial Intelligence if the solution can be achieved with classical or vector logic. This is especially true in the fields of design and testing, where vector- logic mechanisms quickly solve combinatorial problems, simulate faults, and generate minimal test sets for both functions and structures.

The purpose of this study is to significantly reduce the time needed to test the logical functionality of SoCs controlling autonomous robotic objects by using vector logic implemented through in-memory computing. This method removes the need for simulation algorithms and processor instructions. The research objectives are: i) develop methods to create a fault testing map for the logical functionality of the SoC without relying on simulation algorithms. ii) develop methods to generate a fault test map for the bus-interconnect structure of the SoC without using simulation algorithms. iii) Implement and verify applications of vector-logical mechanisms through examples of functionalities and structures. Scientific novelty includes: i) A model that uses exponentially redundant logical vectors to describe and test functions and structures. ii) A model that employs binary addresses of logical vectors to solve combinatorial design and testing problems with linear computational complexity. iii) A mechanism (model plus algorithm equals method) for modeling functional testing maps without simulation or CPU instructions. iv) A mechanism for modeling structure testing maps based on the logical vector of the bus-interconnect description. v) A mechanism for modeling function faults using addresses from truth tables. These mechanisms are unique worldwide for their simplicity and the speed at which maps can be built and functions and structures tested. The study aims to save time and energy by addressing issues in the simulation and testing of functionality, structures, and interconnect buses. This is achieved by modeling vector-logical in-memory computing and using these models to construct test maps. The tasks include: i) Visualizing processes, phenomena, models, and algorithms specified by logical vectors. ii) Modeling a test map for functionalities or structures based on logical vectors to address testing and diagnostic challenges. iii) Modeling a test map for interconnect buses to support system testing and diagnosis of logical faults. iv) Implementing the proposed mechanisms for modeling a test map within a software application and verifying their effectiveness across various examples of structures and functionalities.

2. TRUTH TABLE FAULT SIMULATION

A mechanism for simulating detected faults on the input set using explicit addresses of the truth table is considered. This mechanism is easily scalable to any truth table from n variables. The mechanism can be used to identify faults in test sets of any function or structure. Initial data (Figure 2): logical vector Y, test set T.

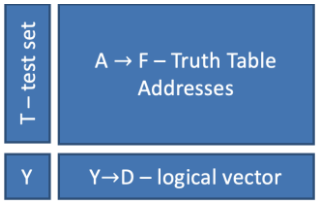


Figure 2. Data structures for fault modeling

Simulation algorithm: i) The output state is calculated as $Y=Y_T$. ii) Vector XOR operations are in progress TY⊕AY to obtain the active input and output coordinates on the test set. iii) Identify faults that are detected in the test set by using the inverse bits of the input set to determine the unit coordinates of the truth table addresses, which are covered by the unit coordinates of the resulting deductive vector. The essence of the modeling mechanism. Identify input faults to be detected by generating a deductive vector for a binary input set. Initial data for modeling (Figure 3): input binary set, length n: T=11 and logical vector Y=0111, length 2^n, formative 2^n truth table input sets X, in the example on variables x_1x_2. Output data: truth table of the faults being detected, dimension $n\times 2^n$. The size of memory to store data structures is determined by the expression: $M = n\times 2^n + 2^n + n+1$, where n is the number of variables of a logical element that forms compact and smart data structures: truth table, logical (deductive) vector, input binary word x=11, and output states Y=1.

Simulation algorithm in detail: i) good-value simulation of logic element –computes the output value Y when feeding a binary input set T, treated as the address of the logic vector bit Y_T. ii) Then, it executes the vector matrix operation L=T⊕F to obtain the activity matrix L. Here the columns, the corresponding variables x_1x_2 treated as binary addresses 11, 01, 10, 00 to recode or reorder the bits of a vector L. iii) Ordering L-Vector coordinates by input binary addresses x_1x_2 truth tables for obtaining a deductive D vector $D_{X_i}=L_i$. This operation is necessary to bring the truth table to the generally accepted standard of incremented binary-decimal addresses based on input variables, which is required for processing elements in a digital structure. Computational complexity of generating a deductive vector: $2^n \times (n + 1)$. 4) Generation of detected input faults of a logic element by inverse values of input variable signals x_1x_2 on the unit coordinates of the truth table columns covered (activated) by the unit values of the deductive vector coordinates: $F_{ij} = \overline{T_i} \leftarrow F_{ij} \wedge D_i =1$. The computational complexity of the algorithm is determined by the following formula: $Q = 1+ 2^n \times (n + 1)+ 2^n+ 2^n \times n$ performing read-write transactions. It is proposed to use a truth table to simulate input faults for any logic element. Simulation formula: the input binary set used as the cell address of the logical vector contains the output state of the component that is concatenated with the input set to form a good state vector designed to xor-interact with all columns of the truth table to create an unordered by addresses of the table and the activity vector to which the procedure of ordering the columns of the table and the bits of the activity vector by standard addresses in corresponding columns of the table, then the resulting deductive vector activates unit coordinates in the corresponding columns of the truth table with its unit values, the signs of which are determined by the inverse states of the input set bit.

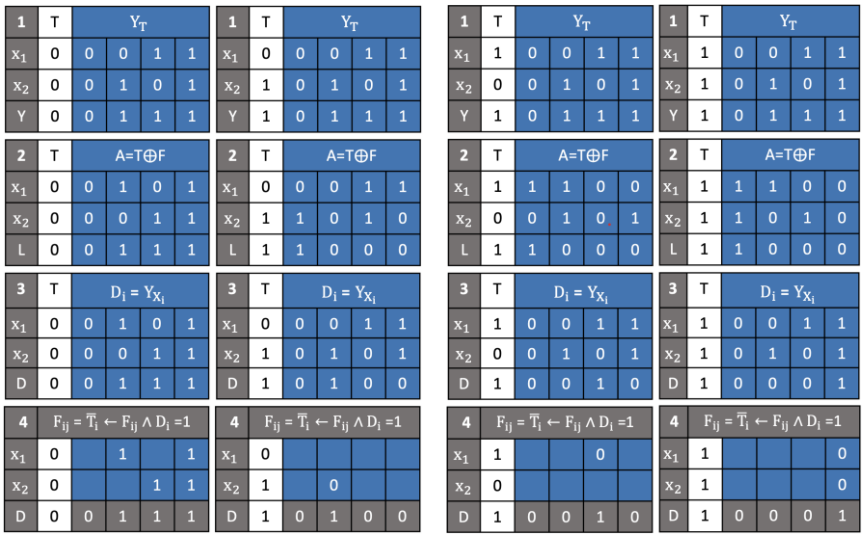


Figure 3. Modeling or function faults on a comprehensive test

3. MODELING A TEST MAP OF A FUNCTION OR STRUCTURE

A modeling mechanism for prompt engineering is proposed to create a test map for all combinations of faults on a logical vector, which acts as a model of a function or structure. This approach employs three matrix operations to represent all potential fault combinations detectable during a comprehensive functionality test involving n variables. A non-traditional logic testing method, based on the use of a logical

constant presented as an H-matrix of faults, is considered (Figure 4). The latter interacts with the activity logic matrix to produce a test map in the form of a ratio of the exhaustive test to all input fault combinations to be tested. The challenges of modeling faults in logical functionality are addressed through exhaustive testing without relying on a specific modeling algorithm.

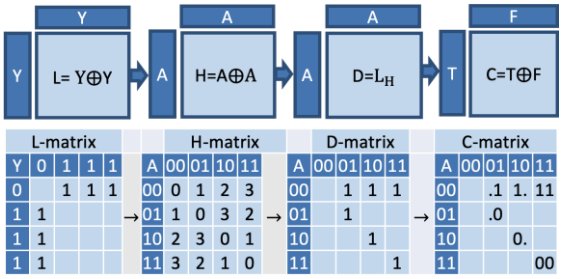


Figure 4. Structure of matrix operations for modeling a test map

The synthesis of the test map of logical functions from three variables given by the vector 11100111 is shown in Figure 5. The procedures for modeling an L-matrix using the Cartesian XOR operation on a logical vector are shown here. An H-matrix is then used to obtain a deductive matrix by recoding the coordinates of the L-matrix according to the formula $D=L_H$. Next, the vectors of the deductive matrix are used to obtain a test map without a fault simulation algorithm. In addition, deductive vectors are used to simulate faults in elements of a logical circuit. It should be noted that the columns of the deductive matrix are all kinds of Boolean derivatives of combinations of input variables. These derivatives can be used to build a minimum test relative to the ordered fault class. A test map is used to model a minimum test relative to single constant faults of input variables. It can also be used to diagnose logical defects in input variables. Figure 5 shows a structural fragment of the formation of the signs of the faults to be detected in the alphabet {0,1,"."}.} at the coordinates of the test map, according to the 1-coordinates of the deductive matrix.

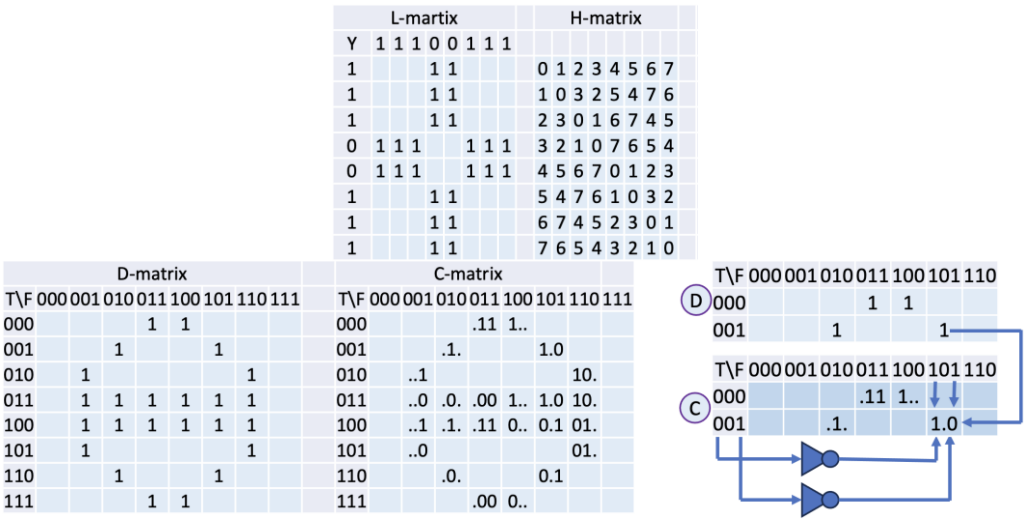


Figure 5. Modelling faults for 11100111-function on exhaustive test

The faults to be detected are generated only by the unit coordinates of the addresses of the truth table F (the top line of the D or C-matrix). These 1-coordinates are always further defined by the inverse bit values of the binary test set bits. The 0-coordinates of the addresses in the test map's truth table are further determined by points, indicating that no faults can be detected on these input variables. In short, the functionality testing map is defined by the Cartesian XOR interaction of the truth table addresses at the 1-coordinates of the deductive matrix. The two truth tables are compact models of the exhaustive test T and all logical faults of the input variables F. Signs of the faults to be detected are formed only from the single

coordinates of the addresses of the truth table F. Zero coordinates of the addresses F form "." points that mean the absence of detected faults on the input variables. Modeling Test Map allows you to process IP Core SoC functionality to obtain a minimum test, evaluate the quality of an existing test, or diagnose input faults without a simulation algorithm. This technology is innovative and can verify the logic of an SoC IP core containing up to 16 variables.

4. LOGIC VECTOR FORMS

A logical vector is a sequence of bits with a length of 2^n, n – a positive integer to describe size of functions and structures. Computing logic consists of four forms: a logical vector, a matrix, a truth table, and a graph in Figure 6. The latter form is exclusively oriented towards the human eye. A graph, like an image, is an attribute of cave computing.

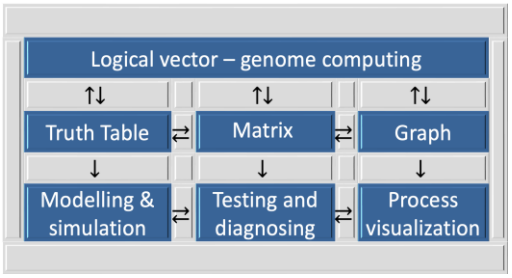


Figure 6. Forms of computing logic

In any computer, a graph is represented by a vector, matrix, or truth table. However, for verifying a process or phenomenon model, a graph serves as a visual proof of the model's or algorithm's validity in human decision-making. The following mechanism constructs graphs from logical vectors by creating matrices corresponding to each vector or truth table. Base: Any logical vector of length 2^n is a matrix of dimension $2^m \times 2^{n-m}$(m≤, n≥0). Any matrix represents a graph using its 1-coordinates, which define the arcs between the vertical and horizontal matrix addresses that serve as identifiers of the graph's vertices. The dimension of the matrix over the total number of coordinates is always a multiple of the power of two (2^n, n=0,1,2,3,4,5,6,7, …): 1, 2, 4, 8, 16, 32, 64, 128, … . The first two vectors, 1 and 2 bits long, form truth tables that do not have all combinations of addresses by columns or rows of the matrix. These are degenerate truth tables that are built on degenerate logical vectors, the length of which is equal to 2^n, n=0, 1. Therefore, we can say that not only is a pair of bits a vector $2^{n=1}$, but also 1 bit according to the definition $2^{n=0}$ is a vector too. Naturally, these degenerate vectors from matrices of dimension 1×1 and 1× 2. Graphs for such vectors are also not distinguished by their complexity. They are one or two vertices on which 0, 1, and 2 arcs are defined, respectively. Hereinafter, the results of an application that implements graph visualization mechanisms for a logical vector are used. Other graphs for nontrivial logical vectors are of two types. These are Cartesian (orthogonal or graphs) and bi-graphs for quadratic matrices, and only bi-graphs for rectangular matrices. The logical vector is the Computing genome. Why? i) It effectively and simply describes the functions and structures in any memory. ii) It generates smart data structures that are focused on latency-energy solution of all tasks, including design and test in-memory. iii) It does not require synthesis and CPU instructions but solves all modeling for simulation problems using read-write transactions on memory. iv) A logical vector creates a visual bipartite-graph image of neural networks, which in a computer takes the matrix form of a logical vector. Logical vector is the cycle of logic in computing: logical vector – truth table – matrix – graph – logical vector. A logical vector is an exhaustive form of logic (functions and structures). Logic Vector serves as a computing gene, encompassing all forms of logic vectors (truth tables, graph matrices) to describe functions and structures, as well as derivatives (activity matrices, recoding matrices, deduction matrices, and matrix (map) testing) to solve all computing problems. Logic is a fundamental aspect of the existence of nature and society. The logical vector equivalents the concepts of function and structure. A bipartite graph is a structure that does not have reflexive and symmetric relations on two non-intersecting sets of vertices. A Cartesian graph is a structure that has reflexive and symmetric relations. The matrix of a Cartesian graph is always quadratic. The matrix of a bipartite (bi-graph) graph can be quadratic or rectangular. The number of coordinates of a quadratic or rectangular matrix is always equal to 2^n. Interest is the graphs of logical functions that form bi-graphs for bus data structures.

5. VECTOR LOGIC MODELING OF THE BUS TEST MAP

Create a test map for a bus with two interconnects in Figure 7. The initial information is the bi-graph of the two connecting buses. A bipartite graph is used to build a matrix with four components: two sets of vertices that define the connecting wires between them. At the same time, each arc has the same vertex identifiers, which encode the beginning and end of the connecting equipotential line in any digital circuit. The resulting matrix is used to write the graph vector, line by line. This vector is then used to build a C test map in three matrix operations. The first operation constructs an A-matrix of activities by taking the Cartesian product of the two sets. Then, using the H-matrix of recoding to obtain a deductive D-matrix, the coordinates of which are determined by the HDL standards of the addresses of the truth table. Further, all unit coordinates of the D-matrix are determined by the vectors of the input faults to be detected. Each significant fault is the inverse value of the corresponding bit of the input test set in place of the 1-coordinates of the fault addresses (at the top of the matrix) of the truth table. The "."-symbol indicates undetected faults. Thus, each fault vector being tested is a projection of the unit coordinate of the D-matrix on the axis of the test suites (the addresses on the left side of the matrix) and the unit coordinates of the addresses of the fault truth table (at the top of the matrix). For a graph, such logic faults mean two options: i) The existence of an arc from an actual vertex to a false one, or ii) the existence of an arc from a false vertex to an actual one.

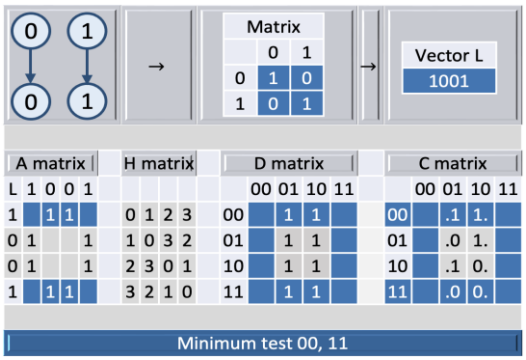


Figure 7. Modeling a test map for a bus with two interconnects

Modeling the bus test map using logical bus vectors is based on matrix mechanisms that utilize Cartesian logic, enabling the construction of a test map for any structure or functionality in three procedures. For the vector 1000010000100001, which is a four-interconnect bus model, the modeling of the test map looks like this in Figure 8. How can I determine the combination of faults tested in the test set for the corresponding cell in the test map? For example, what does the combination of faults 1.11 mean in a cell defined by coordinates 0000-1011? This means that a multiple fault will be detected on test set 0000, which turns arc 00-00 into a non-existent connection 10-11. Visualization of processes and phenomena enables the researcher to gain deeper insight into hidden features of computer models. For example, a bigraph enables modeling bus interconnect tests through a straightforward procedure: building a vector and using it to simulate and test all possible logic-fault combinations. The preparatory procedure involves representing the vertices of the bigraph using binary addresses and encoding the connections between them. The peculiarity here lies in the coding of the sinks and sources of arcs, which form one-to-one correspondences with the same binary codes. Such coding corresponds to the logic of the interconnect, which is an equipotential line of a digital product.

Next comes the Modeling of the matrix of such a bigraph and its conversion into a logical vector. The latter is a prompt engineering request for a tire test map. A test map is obtained from three matrix operations that cover all possible combinations of logic faults on 2 to the n. In this case, a matrix is generated to test all combinations of logical faults on a bus with a dimension of $2^n \times 2^n$. The scientific novelty of this mechanism lies in modeling a test map of the wire bus using the object's logical vector from its structural model. The resulting test map contains some redundant strings. For bus testing, it is always necessary to select those n lines that correspond to the concatenation of the addresses of the beginning and end of the graph arc that form the models of n interconnects in the matrix, with the dimension $2^n \times 2^n$. In this case, we are referring to the following lines on the test map: 0000, 0101, 1010, 1111. It should be noted that the application used to build the test map is in the public domain [29]. The application mechanisms form all the auxiliary matrices and the bus test map. Data bus processing statistics with 8 interconnects. A complete test map contains all combinations of faults that are tested on the minimum test, the number of sets of which is

always equal to the number of n interconnects in the bus. The number of fault combinations that are detected by the minimum test of eight sets is $2^n - n$ (256-8=248). Columns (8), the numbers of which appear in the test, do not contain the faults to be detected. Other lines in the test matrix contain combinations of faults that are detected by sets from the minimum test. The purpose of these lines of the test matrix is to more accurately diagnose defects in the bus interconnects using the full test map. How to decipher the combination of faults in the cell of the test map being tested on the test sets? For example, what does a combination (... 000) of faults mean in the cell defined by the coordinates 111111-000111? This means that a multiple fault will be detected on the test set 111111, which turns arc 000-111 into a connection 000-000.

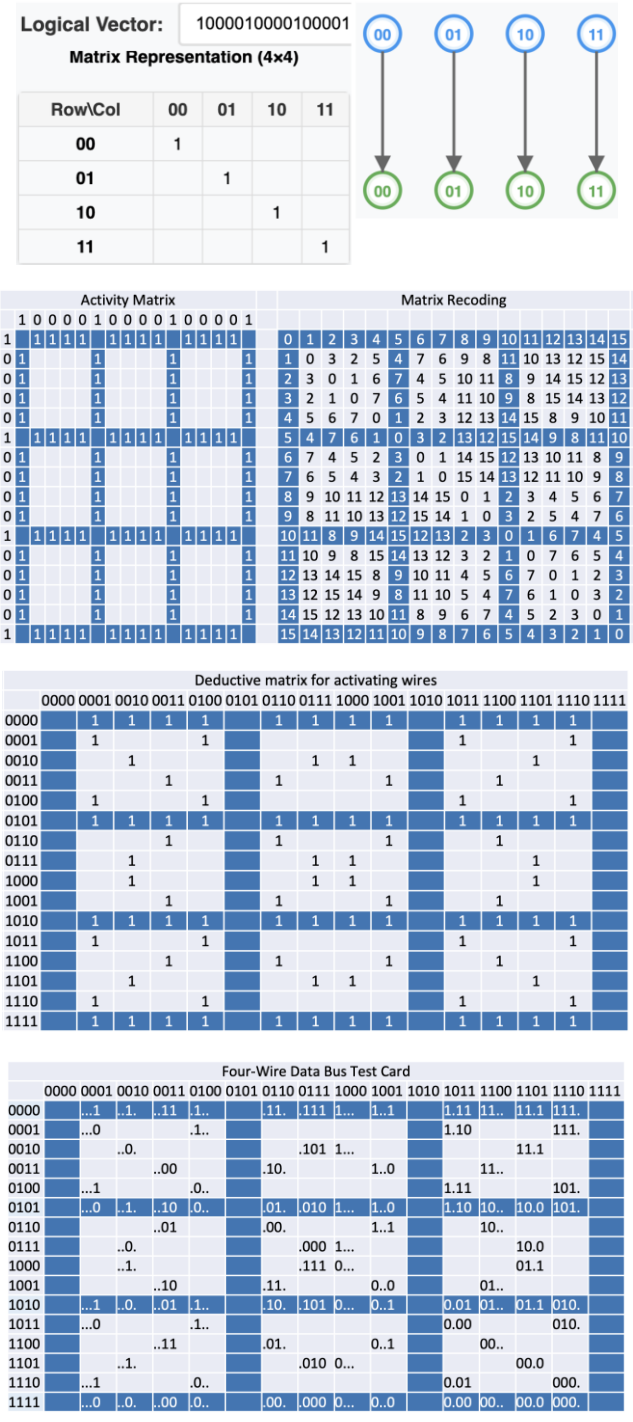


Figure 8. Modelling test map for a four-wire bus

6. DISCUSSION OF THE RESULTS OF THE STUDY

The modelling for simulation architecture [29], [30] contains the following services: i) Modelling of digital circuit data structures for good-value simulation. ii) Modelling data structures for simulation faults as addresses on test sets. iii) Modelling of a logical vector {circuits, structures, graphs, buses} for building a test map based on prompt engineering technology. iv) Modeling smart mechanisms (activity matrix, distance matrix, deductive matrix, test map) to perform modeling for the simulation of any process or phenomenon. v) Modeling a graph along a logical vector for visualizing a model of a process or phenomenon. vi) Modeling interfaces for entering circuit or functionality: GUI, HDL, Python to model internal smart data structures to simulate test sets and faults like addresses. vii) Modeling tables of faults to be detected on test sets by simulating them as addresses to build a minimum logical circuit test. viii) Modelling of the map for testing the functionality specified by the logical vector using the technology of prompt engineering for three matrix operations. ix) Modelling data structures to perform fault diagnosis based on the test map. x) Modelling of input and output data structures at the request of the user. Some statistics for processing logical vectors for building a test map for functionalities, structures, and interconnect buses are presented in Figure 9.

Latency of the modeling test map					
Function, n=	0	1	2	3	4
ms=	0,10	0,10	0,10	0,20	0,30
Continued	5	6	7	8	9
ms=	0,60	2,20	10,00	24,00	105,80
Structure, n=	0	1	2	3	4
	0,10	0,10	0,10	0,30	0,30
Continued	5	6	7	8	9
	0,70	2,00	12,00	24,00	101,80
Bus, n=	0	1	2	3	4
	-	-	0,10	0,30	0,30
Continued	5	6	7	8	9
	0,60	2,40	16,00	26,00	115,00

Figure 9. Latency of the modeling test map

Here, parameter n denotes the number of variables that form the length 2^n of the vector to model the test map. Practical significance of the work: i) Simulation of faults on a full verification test or construction of a test map on a logical vector takes 3 matrix operations. Simulating the faults of an equivalent logic circuit during a full validation test will take an order of magnitude longer. ii) Similar latency estimates exist for good-value simulation. Using a logic vector instead of a circuit speeds up the simulation results by an order of magnitude. iii) Constructing a minimum test by minimizing an exhaustive test of a test card where the faults to be tested are visible on each set is much more technologically advanced than using circuit fault simulation on test kits with unknown fault simulation results. iv) Constructing a conductor test map involves splitting all bus-interconnects into segments and then using a logical vector to build a test map for each segment. v) Disadvantages: i) vector-logical modeling consists of the screen rendering of the test map for logical vectors having more than 16 variables; ii) the technological complexity of representing elementary automata by a logical vector, such as registers, counters, triggers, and circuits with feedback. Software applications based on vector logic for mapping testing functionalities and structures can be used in the educational process for computer engineering specialties. The proposed cloud services can be used to verify projects related to the creation of robot control systems, where logic circuits integrated into SoCs running the IEEE 1500 SECT standard dominate [27]. In contrast to good heuristics and algorithms, chip-let interconnect testing [20]–[22] offers a simple system solution based on logical vectors. A bigraph is represented as a matrix, which is then transformed into a logical vector by its rows to build a test map through three matrix operations. A minimum test of any interconnect bus, the length of which is always equal to the number of interconnects in the bus. The remaining test map is necessary for accurately diagnosing all possible logic faults in the bus wires, which are features of the graph model. The vertices of the start and end of each arc have the same codes when modelling the bigraph. Testing path: bigraph – matrix – logical vector – testing map. Visualization path: logical vector – matrix – graph. Rendering here serves as an amplifier of understanding the essence of a process or phenomenon, enabling a person to develop an accurate computer model. Modelling and rendering harmoniously complement each other. A logical vector is a structure that solves the problems of computer modeling and human rendering.

7. CONCLUSION

Vector-logical mechanisms for modeling a test map of functionalities, structures, and buses without a simulation algorithm are presented. The logic vector emphasizes solving design and verification problems by leveraging cost-effective in-memory computing based on read-write transactions without processor instructions. A logical vector serves as a universal model that addresses the challenges of modeling and simulation for functions, structures, graphs, and data buses. Vector logic mechanisms are used to construct test maps, graphs, and functionalities. All logical operations can be performed on vector graph models to produce results in the form of a logical vector, matrix, or graph, including functionalities, structures, and interconnect buses. A logical vector can model any process or phenomenon, and it can also be used to diagnose errors and faults. It is a structure that solves the problems of modeling for a computer and visualization for a human. The mechanisms and applications presented here may interest students, scientists, and researchers involved in digitizing processes and phenomena for modeling and verification.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT *(mandatory)*

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Vladimir Hahanov	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Svetlana Chumachenko					✓	✓			✓	✓		✓	✓	
Eugenia Litvinova						✓			✓	✓				
Andrii Voronov			✓			✓				✓				
Oleh Demchenko			✓			✓				✓				
Nataliya Maksymova				✓		✓				✓				

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

- The data that support the findings of this study are available on request from the corresponding author, [Vladimir Hahanov].
- Derived data supporting the findings of this study are available from the corresponding author [Vladimir Hahanov] on request.
- The authors confirm that the data supporting the findings of this study are available within the article.

REFERENCES

[1] N. Darabi, P. Shukla, D. Jayasuriya, D. Kumar, A. C. Stutts, and A. R. Trivedi, "Navigating the unknown: uncertainty-aware compute-in-memory autonomy of edge robotics," in *Proceedings -Design, Automation and Test in Europe, DATE*, 2024, pp. 1–6, doi: 10.23919/date58400.2024.10546628.

[2] Y. Xia, N. Jazdi, J. Zhang, C. Shah, and M. Weyrich, "Control industrial automation system with large language model agents," in *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, 2025, pp. 1–8, doi: 10.1109/ETFA65518.2025.11205539.

[3] S. Mahmood Mahdi, A. I. Abdulkareem, A. Jaleel Humaidi, A. K. Al Mhdawi, and H. Al-Raweshidy, "Comprehensive review of control techniques for various mechanisms of parallel robots," *IEEE Access*, vol. 13, pp. 63381–63416, 2025, doi: 10.1109/ACCESS.2025.3557937.

[4] K. Deng *et al.*, "System-on-chip test and characterization: a review," *IEEE Transactions on Instrumentation and Measurement*, vol. 74, pp. 1–28, 2025, doi: 10.1109/TIM.2025.3584134.

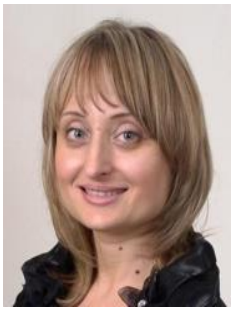
[5] N. R. Shanbhag and S. K. Roy, "Benchmarking in-memory computing architectures," *IEEE Open Journal of the Solid-State Circuits Society*, vol. 2, pp. 288–300, 2022, doi: 10.1109/OJSSCS.2022.3210152.

[6] V. Hahanov, S. Chumachenko, E. Litvinova, I. Hahanov, Z. Davitadze, and D. Devadze, "Vector logic of computing," in *2025*

IEEE 13th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), 2026, pp. 1–5, doi: 10.1109/idaacs68557.2025.11322309.
[7] T. Sharma, I. Chakraborty, M. Ali, and K. Roy, "Evaluating compute in memory architectures for matrix multiplication: a dataflow-centric perspective," in *2025 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2025, pp. 1–3, doi: 10.1109/ispass64960.2025.00056.
[8] H. Guo, J. Chen, and H. Jiao, "A 15T SRAM cell-based fully-digital computing-in-memory macro supporting high parallelism and fine-grained simultaneous read + write + MAC operations," *IEEE Solid-State Circuits Letters*, vol. 8, pp. 153–156, 2025, doi: 10.1109/LSSC.2025.3567840.
[9] S. Zhang, X. Cui, F. Wei, and X. Cui, "An area-efficient in-memory implementation method of arbitrary boolean function based on SRAM array," *IEEE Transactions on Computers*, vol. 72, no. 12, pp. 3416–3430, 2023, doi: 10.1109/TC.2023.3301156.
[10] H. Lee, J. Lee, and S. Kang, "An efficient test architecture using hybrid built-in self-test for processing-in-memory," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 33, no. 5, pp. 1452–1456, 2025, doi: 10.1109/TVLSI.2024.3504539.
[11] J. Yin *et al.*, "A dexterous robotic hand with cost-effective structure-function integrated tactile fingertip for enhanced grasping and robotic palpation," *IEEE Transactions on Instrumentation and Measurement*, vol. 74, pp. 1–16, 2025, doi: 10.1109/TIM.2025.3635800.
[12] M. Haug *et al.*, "Structure-function relationships in muscle fibres: *MyoRobot* online assessment of muscle fibre elasticity and sarcomere length distributions," *IEEE Transactions on Biomedical Engineering*, vol. 69, no. 1, pp. 148–155, 2022, doi: 10.1109/TBME.2021.3089739.
[13] M. Davis, "Emil Post's contributions to computer science," in *Proceedings. Fourth Annual Symposium on Logic in Computer Science*, 1989, pp. 134–136, doi: 10.1109/lics.1989.39167.
[14] W. Liu, Q. Zhao, and Q. Zhang, "Construction of vectorial boolean functions with provable differential-linear uniformity and high nonlinearity," in *2024 6th International Conference on Natural Language Processing, ICNLP 2024*, 2024, pp. 300–304, doi: 10.1109/ICNLP60986.2024.10692619.
[15] S. Zhu *et al.*, *Intelligent computing: The latest advances, challenges, and future*, vol. 2. 2023.
[16] K. Liu, J. Wu, H. Yang, Q. Sun, and R. Wan, "SRTEF: Test function recommendation with scenarios and latent semantic for implementing stepwise test case," *IEEE Transactions on Reliability*, vol. 71, no. 2, pp. 1127–1140, Jun. 2022, doi: 10.1109/TR.2022.3164645.
[17] A. Simion, "Test-driven development in Go: a practical guide for writing idiomatic and efficient Go tests through real-world examples," p. 390, 2023.
[18] G. Chandrasekaran, N. S. Kumar, P. R. Karthikeyan, K. Vanchinathan, N. Priyadarshi, and B. Twala, "Test scheduling and test time minimization of system-on-chip using modified BAT algorithm," *IEEE Access*, vol. 10, pp. 126199–126216, 2022, doi: 10.1109/ACCESS.2022.3224924.
[19] V. Hahanov, W. Gharibi, S. Chumachenko, and E. Litvinova, "Vector synthesis of fault testing map for logic," *IAES International Journal of Robotics and Automation*, vol. 13, no. 3, pp. 293–306, 2024, doi: 10.11591/ijra.v13i3.pp293-306.
[20] IEEE, "IEEE standard for a mixed-signal test bus," *IEEE Std 1149.4-2010 (Revision of IEEE Std 1149.4-1999*, vol. 2010, no. March. pp. 1–116, Dec. 2011, doi: 10.1109/IEEESTD.2024.10776765.
[21] J. Wang, Z. Liu, J. Wang, M. Wang, B. Gu, and Y. Cheng, "Optimal sequential-parallel test strategy generation method for complex systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 55, no. 7, pp. 5054–5068, 2025, doi: 10.1109/TSMC.2025.3560997.
[22] X. Bai and S. Tan, "An efficient surface-integral-equation based Nyström method with an over-determined testing scheme for broadband grating scattering modeling," *IEEE Journal on Multiscale and Multiphysics Computational Techniques*, vol. 10, pp. 125–136, 2025, doi: 10.1109/JMMCT.2025.3535936.
[23] J. G. Pauloski, K. Chard, and I. Foster, "Agentic discovery: closing the loop with cooperative agents," *Computer*, vol. 58, no. 10, pp. 20–27, 2025, doi: 10.1109/MC.2025.3575029.
[24] R. Shan, "AI that learns, thinks, and acts: the next frontier of generative AI," *Computer*, vol. 58, no. 10, pp. 28–39, 2025, doi: 10.1109/MC.2025.3582568.
[25] Y. Qin, Y. Zhao, and L. Zhao, "Exploring the ideas of meta-learning and knowledge distillation: core concepts and future development trends of large models," *Computer*, vol. 58, no. 10, pp. 40–48, 2025, doi: 10.1109/MC.2025.3574676.
[26] A. Chandra, E. Garita-Rodriguez, and P. Ghosh, "Test, debug, and repair for chiplet-based designs," in *2025 IEEE 43rd VLSI Test Symposium (VTS)*, 2025, pp. 1–1, doi: 10.1109/vts65138.2025.11022902.
[27] IEEE, "IEEE standard for static component interconnection test protocol and architecture," *IEEE Std 1581-2011*, no. June. pp. 1–61, Sep. 2011.
[28] S. Das, F. Su, and S. Chakravarty, "Testing of prebond through silicon vias," *IEEE Design and Test*, vol. 37, no. 4, pp. 27–34, 2020, doi: 10.1109/MDAT.2020.2968255.
[29] "Vector simulation model." https://vector-simulation-model.vercel.app/ (accessed Feb. 09, 2026).
[30] "GenLV vector to graph." https://andreyv99.github.io/GenLV/vector-to-graph (accessed Feb. 09, 2026).

BIOGRAPHIES OF AUTHORS

Vladimir Hahanov was born in the USSR in 1953. He is a doctor of science and professor in the Faculty of Computer Engineering, Department of Design Automation, Kharkiv National University of Radio Electronics, Ukraine. His research interests include computer design and testing, test generation and fault simulation for SoCs, quantum memory-driven and vector-logic computing, cyber-physical and cyber-social computing, pattern recognition and machine learning, digital smart cyber universities, and cloud-driven traffic control. He has supervised four doctors of science and 36 Ph.D. students. Since 2003, he has served as the general chair of the IEEE East-West Design & Test Symposium. He is the author of more than 650 publications, 25 textbooks, and 5 patents and has published 212 Scopus-indexed papers with 1,209 citations. His h-index is 16. Prof. Hahanov has been an IEEE Senior Member since 2010 and is also a member of the IEEE Computer Society Golden Core, SAE International, IFAC, and the IEEE Communications Society. He can be contacted at vladimir.hahanov@nure.ua.



Svetlana Chumachenko was born in the USSR in 1969. She is a doctor of technical sciences, professor, and head of the Department of Design Automation at Kharkiv National University of Radio Electronics, Ukraine. Her research interests include mathematics, computer engineering, and smart cyber universities. Her international activities include participation in the scientific and technical cooperation agreement "Strategic Partnership" with Aldec Inc. (USA) in 2000 and 2005; the SEIDA BAITSE (Baltic Academic IT Security Exchange) project at Blekinge Institute of Technology, Sweden, during 2011–2012; and the international project Curricula Development for New Specialization: Master of Engineering in Microsystems Design (530785-TEMPUS-1-2012-1-PL-TEMPUS-JPCR) under the Curricula Reform priority during 2012–2016. She is the author of more than 250 publications, 5 textbooks, and 109 Scopus-indexed papers with 518 citations. Her h-index is 12. She can be contacted at svetlana.chumachenko@nure.ua.

Eugenia Litvinova was born in Ukraine in 1962. She is a doctor of science and professor in the Faculty of Computer Engineering, Department of Design Automation, Kharkiv National University of Radio Electronics, Ukraine. Her research interests include computer design and testing and quantum computing. Her international activities include serving as a staff member of the TEMPUS Project No. 530785-TEMPUS-1-2012-PL-TEMPUS-JPCR, Curricula Development for New Specialization: Master of Engineering in Microsystems Design, and as a member of the organizing committee of the IEEE East-West Design & Test Symposium since 2007. She is the author of more than 250 publications, 5 textbooks, 3 patents, and 116 Scopus-indexed papers with 584 citations. Her h-index is 12. She can be contacted at eugenia.litvinova@nure.ua or litvinova_eugenia@icloud.com.

Andrii Voronov is a Ph.D. student in the Department of Design Automation, Faculty of Computer Engineering, Kharkiv National University of Radio Electronics, Ukraine. His research interests include cyber-physical computing, SoC design and testing, and machine learning. He can be contacted at andrii.voronov@nure.ua

Oleh Demchenko is a Ph.D. student in the Department of Design Automation, Faculty of Computer Engineering, Kharkiv National University of Radio Electronics, Ukraine. His research interests include cyber-physical computing, SoC design and testing, and machine learning. He can be contacted at oleh.demchenko@nure.ua.

Nataliya Maksymova was born in Ukraine. She received a master's degree in computer science from Kharkiv National University of Radio Electronics, Ukraine, which was evaluated by the University of Toronto as equivalent to a Canadian master's degree. She is currently a professor at Niagara University, Vaughan Campus, Ontario, Canada. Her research interests include SoC design and testing and machine learning. She can be contacted at nmaximova@hotmail.com.