DOI: https://doi.org/10.15276/aait.08.2025.28
UDC 004.582

Vector logic structures and functions of computing

Vladimir I. Hahanov[1)]
ORCID: https://orcid.org/0000-0001-5312-5841; hahanov@icloud.com. Scopus Author ID: 7801667873
Svetlana V. Chumachenko[1)]
ORCID: https://orcid.org/0000-0001-8913-1194; svetlana.chumachenko@nure.ua. Scopus Author ID: 57188710840
Eugenia I. Litvinova[1)]
ORCID: https://orcid.org/0000-0002-9797-5271; eugenia.litvinova@nure.ua. Scopus Author ID: 25650378900
Hanna V. Hahanova[1)]
ORCID: https://orcid.org/0000-0002-4528-6861; anna.hahanova@nure.ua. Scopus Author ID: 8326375900
Volodymyr I. Obrizan[1)]
ORCID: https://orcid.org/0000-0002-1835-4056; vladimir.obrizan@nure.ua. Scopus Author ID: 15127546800
Nataliya H. Maksymova[1)]
ORCID: https://orcid.org/0009-0006-0293-655X; nataliya.maksymova@nure.ua. Scopus Author ID: 59669839800
[1)] Kharkiv National University of Radio Electronics, 14, Nauky Ave. Kharkiv, 61166, Ukraine

ABSTRACT

Vector logic computing is a cost-effective mechanism for intelligent in-memory computing, utilizing read-write transactions to address practical problems in the analysis and management of physical, social, and business processes based on monitoring. The concept of a mechanism is introduced as a harmonious relationship between the model and the algorithm for computing. The investigation aims to significantly reduce time and energy costs associated with simulation processes in the physical, social, and digital worlds. The means to achieve this goal is using vector-logical in-memory computing mechanisms, which significantly simplify algorithms due to the exponential redundancy of data structures. The mechanisms for reducing the algorithm's computational complexity, which is typically exponential due to the use of data structures based on vector logic, are considered. An analysis of functions and structures is conducted from the perspective of algorithmic simplicity for modeling and simulation purposes. Several mechanisms based on Cartesian logic are proposed for modeling a logical vector from the structure of a digital circuit. A Boolean vector is used to create a test map in three matrix operations. A cloud-based MOSI service is offered that simulates the behavior and faults of digital circuits and their functionalities using truth table addresses.

Keywords: Element structure; vector logic functionality; vector logics; logic vector; Cartesian logic; intelligent computing; test map; fault simulation; good-value simulation; in-memory modeling for simulation; prompt engineering

For citation: **Hahanov V. I., Chumachenko S. V., Litvinova E. I., Hahanova H. V., Obrizan V. I., Maksymova N. H. "Vector logic structures and functions of computing". *Applied Aspects of Information Technology*. 2025; Vol.8 No.4: 442–452. DOI: https://doi.org/10.15276/aait.08.2025.28**

METRIC OF COMPUTING STRUCTURES AND FUNCTIONS

Structures and functions are the most common forms of the model, which, together with algorithms for their analysis, make up the mechanisms of modern computing. Their harmonious relations are considered to replace one form with another, thereby saving time and energy in organizing the computational process in memory. Historically, the digital circuit emerged when it was necessary to synthesize a high-speed combinational circuit from a truth table, given the need to save slow but expensive memory. Memory today is an affordable, cheap, energy-saving product, the speed of which is measured in the nanosecond range. Why synthesize a logic circuit when a logic vector in memory can represent functionality to organize calculations without processor instructions? In addition, the logic circuit not only requires expensive synthesis procedures but also presents algorithmic difficulties for its modeling, testing, verification, and diagnosis. What is offered instead is to return to the truth table (logical vector), which does not require synthesis, but only needs to perform modeling procedures in any memory. At the same time, all verification problems are solved on the logical vector by algorithms of linear computational complexity due to the exponential redundancy of the model. The question remains what to do with the many digital circuits that exist in cyberspace. You can re-engineer them by transforming the circuit into a functionality represented by a logical vector and then solving all verification problems using linear-complexity algorithms. Several mechanisms for constructing a logical vector for the combinational structure of elements are proposed.

The results of building a functional testing map using the constructed logical vector of the scheme are also presented. The MOSI software application is open to students and professionals who want to learn about prompt engineering in solving design and test problems. The GUI enables you to create a circuit on a 20-element screen in 9 minutes, which is a very technologically advanced and cost-effective approach when students conduct a laboratory workshop. Modelling of the logical vector of a combinational circuit for good value and faults, as addressed, is proposed for simulation in the in-memory computing architecture. The logical vector is the most technological, compact and comprehensive representation of the scheme for the economical solution of all design and test tasks Cartesian logic is proposed, which, due to exponential redundancy, 2^{n+m}, is an effective intelligent mechanism for solving combinatorial problems (modeling, simulation, testing, diagnostics) by algorithms of linear computational complexity. It is a logical vector (matrix) because it models Cartesian logical relations between bits of logical vectors or addresses in a truth table.

Cartesian logic addresses the following issues:

1) Modeling circuit logic vectors without an algorithm that ensures good behavior.

2) Fault modeling testing map of logic without a fault modeling algorithm.

The article discusses the issues of reducing the cost and time of verifying digital projects by modeling the logical vector of a digital circuit, which enables the significant simplification of value-driven simulation algorithms and reduces the synthesis of a test map to three matrix operations.

The practical significance of the study lies in an economical (in terms of time and energy) vector-logical solution to design and test problems in the architecture of in-memory computing based on read-write transactions, eliminating the need for processor instructions. The theoretical material can be helpful for engineers and students to understand the processes of modeling and simulation in vector-logical verification of digital projects.

Structures and functions [1], [2], [3] are the most common forms of the model, which, together with the algorithms for their analysis, make up the computing mechanisms. A logical vector is a universal form for specifying structures and functions, which also allows you to convert these entities into each other. In this case, a logical vector, a truth table, and a matrix are a scalable form of specifying functionality or structure. A logical vector metric is an explicit setting of the combinatorics of all relationships by bit addresses, which allows you to simulate input conditions without processing algorithms. We assume that the world consists only of functions and structures; in this case, the logical vector serves as a universal model for analyzing physical and social processes using known binary modeling methods [3], [4]. The metrics of a function and its structures are their similarities and differences (Fig. 1), which determine the areas of their possible application as components of a computing mechanism [5].

Computing		
Structure	↔	Function, logic
Relationships between (logical) primitives	1	Relationships Between Input and Output Variables
Needs synthesis	2	Needs modeling
Ergonomic for humans	3	Ergonomic for computing
Need an algorithm for analysis	4	No need for algorithm of analysis
Structural depth forms a large delay	5	Latency is determined by a read-write transaction on memory
Complex algorithms for design and test	6	Zero or low algorithms for design and test
The form of the model is a matrix, logical vector, or truth table	7	The form of the model is a matrix, logical vector, or truth table
Combinatorics requires an n^2 algorithms	8	Complete explicit solution combinatorics

Fig. 1. **Metrics of computing structures and functions**

Source: **compiled by the authors**

Analogues of relations between functions and structures are 1) functionality and 2) logical circuits [6], [7], [8]. The first one today does not need any additional processing related to the synthesis of technological elements into a permitted system. It is sufficient to place it in any memory and use it as a model for simulating binary input sets [9], [10], [11], [12]. The second model requires funds and time for its synthesis, as well as processor instructions, hardware description languages, and complex algorithms for verifying synthesized logic circuits [13], [14]. The latency of the resulting logic circuit during its operation can be greater than the write and read operations on memory [15] required using the vector logic model. The lack of a processor for the operation of the vector-logical model makes it economical in terms of time and energy costs. This fact suggests that in-memory computing, combined with vector logic, is the future of mass computing [16], which can be quickly implemented at any point

in space, provided there is material available for organizing memory. Given that a logical vector is a deterministic form of a qubit [14], [15], it is possible to create quantum computers with a long decoherence time to organize memory in the presence of a stable structure of elementary particles. The proposal to abandon logical schemes in favor of logical vectors is a gain in computing time and energy due to the exponential structures of explicit data represented by vectors, truth tables, or matrices [16], [17], [18], [19].

Two problems arise:

1) how to construct a logical vector of a process or phenomenon;

2) how to model [20], [21], [22], [23], [24], [25] a vector according to the existing logical circuit.

The first option involves the unitary encoding of process patterns on the universe of found primitives, with their subsequent entry into the row variables of the truth table. After that, all combinatorial problems related to analyzing big data using the resulting truth table are solved. The second option is to propose several mechanisms for processing digital structures to obtain the logical vector of the scheme [26], [27]. A recursive algorithm for modeling the logical vector of a circuit using Cartesian logic applied to the elements of a digital structure is proposed in this work. Another algorithm is based on the use of a matrix that models each line of the circuit as a set of elements loaded into the simulated output. The third algorithm utilizes the logical vectors of the elements to populate the table of good behavior for all lines in the circuit during a comprehensive test. In this case, the input effects of the circuit elements are considered as the addresses of the bits of the logical vector, which form the state of the processed line using the read-write transaction [27], [28]. As an example for verifying the proposed algorithms, the Schneider circuit is considered which represents a logical structure with converging branches. All the proposed algorithms are implemented in Python code and are made available as a cloud service for free access to students.

The purpose of the study is to reduce the time of verification of a digital circuit by building a logical vector that allows you to create a functional testing map in three matrix operations.

Objectives:

1) development of Cartesian logic mechanisms for constructing a logical vector of the circuit;

2) development of three methods for constructing a logical vector of the scheme based on the use of vector models of elements;

3) simulation of a test map of a logical circuit using the constructed logical vector;

4) verification of the developed mechanisms using the Schneider circuit as an example.

1. MODELING A CIRCUIT LOGICAL VECTOR USING THE CARTESIAN LOGIC

The procedure for constructing a logic vector is illustrated using the example of the Schneider circuit (Fig. 2), which contains reconvergent fan-outs on a four-input element. This can be achieved by converting all elements of the circuit to two-input ones. Therefore, the right side of Fig. 2 shows the decomposition of the last component of the circuit into three elements, two of which are or-elements, and the third element is or-not.

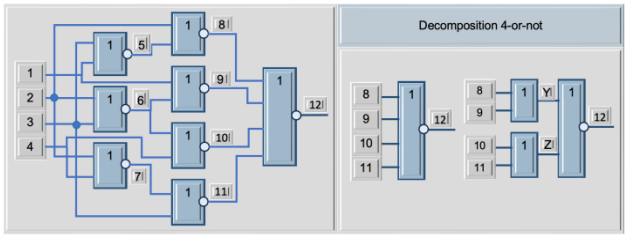

Fig. 2. **Schneider circuit and four-input element decomposition**

Source: **compiled by the authors**

A logical vector is a representation of a process or phenomenon, function, or structure by a sequence of zeros and ones of length 2^n, where n is the number of bits of binary variables to form the addresses of the vector bits. This definition makes the logical vector independent of the truth table. But a mandatory implicit attribute of a vector is the standard addresses, which are easily generated for any vector. The Cartesian product of one vector with itself forms a reflexive (Fig. 3) relation L= Y⊠Y, which generates a Cartesian matrix, in this case, activity, or a logical vector L of dimension 2^{n+1}.

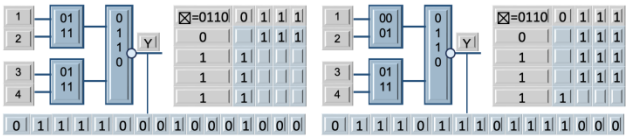

Fig. 3. **Xor-relation of the vectors**.

Source: **compiled by the authors**

Cartesian product of two different vectors 0001⊠0111=0111011101111000 forms or synthesizes a new vector of dimension $2^{n+m} = 2^n \times 2^m$. The circuit vector design of such

Cartesian logic operation is invariant to the length of logical vectors.

All of them are truth tables or logical vectors. Nobody ever needs a digital circuit. It only causes problems – expensive synthesis and complex verification algorithms. All that is required is the output state, as the model's reaction to the input word. All of this can be done in memory, free of a processor, based on read-write transactions over the bits of a logical vector. But you need to get it for this. The complexity of the algorithm (Fig. 4) lies in the recoding of the addresses of truth tables to the numbering of input variables (231–123, 243–234) for the correct arrangement of single values in the logical vector. The last element (four-input) is broken down into three two-input elements, where the first two are or elements. Therefore, in the two Cartesian matrices, the generating function is the logical vector 0111. The computational complexity of the algorithm is quadratic. Model redundancy is also defined by the square of the length of the element's logical vector. Essentially, each circuit element is processed separately, considering the numbering of the circuit line, and then all elements are iteratively bound to the input circuit variables. The algorithm can construct a logical vector for both the entire circuit and its fragments, ending with the outputs of the elements. Build procedures are used only by read-write transactions. Empty cells in the matrices denote zeros. The vectors of the logical elements of the circuit are read into matrices from left to right, from top to bottom, at essential coordinates.

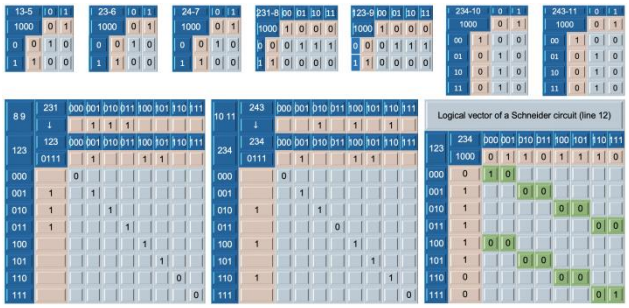

Fig. 4. **Constructing a Logical Vector using Cartesian Logic**

Source: **compiled by the authors**

What are the disadvantages of the proposed algorithm for constructing a logical vector of the circuit:

1) modeling of two-input circuit elements. Logic with many inputs must be decomposed into two-input elements;

2) it is necessary to recode the addresses of the truth table and the bits of the logical vector of each element by the numbers of the external inputs of the circuit;

3) a complex data structure for the execution of the algorithm, which is associated with the construction of a Cartesian matrix for each element.

2. PARALLEL MODELING OF CIRCUIT ELEMENTS

Construction of a logical vector of a circuit using the parallel method: modeling of the outputs of logical elements in one pass through the table of logical vectors of each line of the circuit. The table of dimension $N \times 2^n$, n is the number of input lines of the circuit; N is the number of lines of the circuit. A single line (table column) is modeled in parallel, corresponding to the output of one element on all test cases, and splitting the circuit line modeling table into sub-tables complicates the modeling algorithm, but reduces the total memory for data structures and the number of bit read-write operations. The source data is represented by the circuit structure and logical vectors for each element. The lines of the circuit must be numbered: first, all external inputs are numbered, and then the outputs of elements whose inputs are already numbered; last, the external outputs of the circuit are numbered. The proposed method frees the algorithm from the need to renumber addresses for logical vector coordinates. Here, all the coordinates of all logical vectors have a common addressing on the scale of a single local or global truth table. Here are two options for parallel modeling of the circuit. The first option (Fig. 5) involves decomposing a four-input element by adding an inverter at each of its outputs. In practice, Cartesian logic and local truth tables are used here, which assemble logical vectors for each line of the circuit under the auspices of the external inputs of the circuit. The superposition of logical vectors within local truth tables is an efficient handling of circuit fragments that depend on the same input variables.

The second variant of parallel modeling of logic gates uses a four-input element without decomposition into two-input elements. The data structures of modeling the circuit elements to obtain a logical vector are represented by the modeling table and logical vectors of each circuit element (Fig. 6). There are only two such vectors for this scheme: 1000 and 1000 0000 0000 0000. Empty cells in the table correspond to 0-values. Using the input actions as the addresses of the bits of a logical vector, we determine the reaction of each element to the input action.

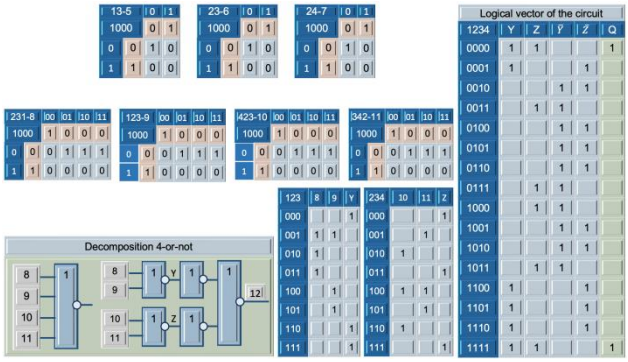

Fig. 5. **Data structures for parallel element modeling**

Source: **compiled by the authors**

In one iteration represented in the column of the modeling ta, we determine all the states of the actual component ble. Eight iterations are required to obtain the logical vector of the circuit.

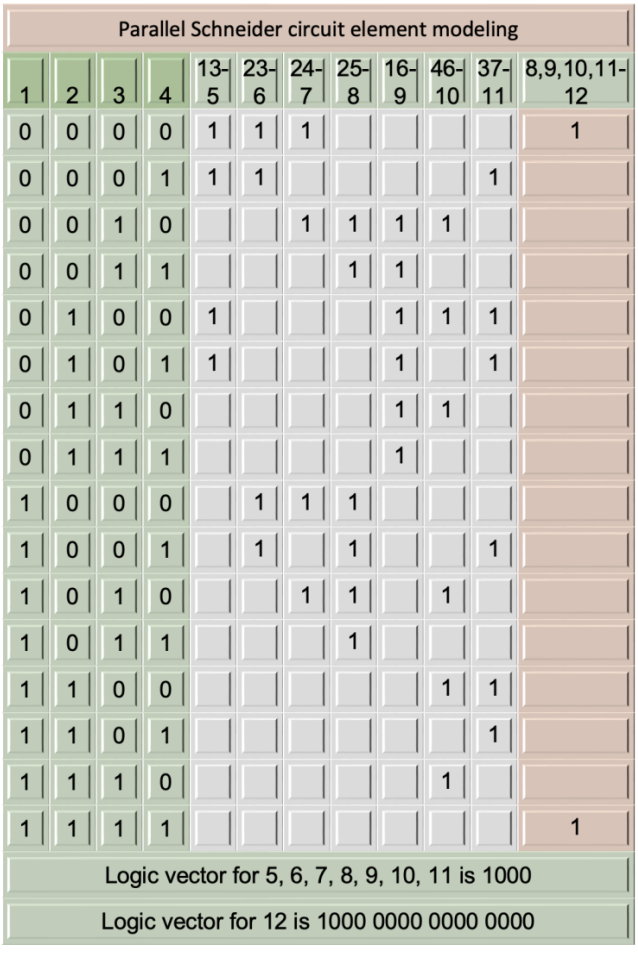

Fig. 6. **Data structures for parallel modeling of elements**

Source: **compiled by the authors**

The method of parallel modeling of input sets as addresses of bits in a logical vector, using a table of all circuit lines, is invariant to the number of inputs of circuit elements. Moreover, increasing the number of inputs on the circuit element increases the simulation speed, which becomes more parallel in a sense. In the extreme, if you represent the circuit as a single logical vector, then modeling the circuit to determine the logical state of the output is performed in a single read-write transaction. This is also a bonus, which is obtained when building a logical vector of the circuit. But the most crucial advantage of the vector is that its use makes the modeling of the test map free from the algorithm (Fig. 7). To build it, you need to perform only two matrix operations, using only the logical vector of the circuit [25], [26], [27]. The resulting test map is a matrix of relationships between test suites and combinations of digital functionality input faults being tested.

Based on this, the following tasks can be addressed:

1) determine the minimum test for single constant faults;

2) identify faults that can be checked on specified test sets;

3) develop schemes for diagnosing faults and testing online functionality;

4) automatically generate a testbench that includes the minimum number of test cases to verify critical functional modes.

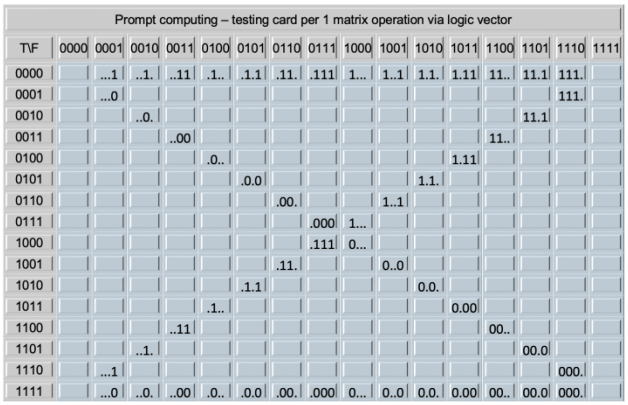

Fig. 7. **Modeling of the test map in two matrix operations**

Source: **compiled by the authors**

In the test map (matrix), undetected faults in input variables are represented by dots, while detected ones are represented by symbols 0 and 1.

The map is constructed through three matrix operations [25], [26], [27]:

1) an XOR operation is performed on the bits of the logical vector to obtain an activity matrix;

2) obtaining a matrix of deductive vectors based on the recoding of the activity matrix from the standard addresses of the truth table;

3) formation of vectors of tested faults in 1-coordinates of the test map. All specified faults are inverses of the bits in the test sets.

3. VECTOR MODELING VS. CIRCUIT SYNTHESIS

Which is better, vector modeling or circuit synthesis? Let us consider three variants of Schneider's circuit (Fig. 8):

1) the original, synthesized heuristically, represented by eight components and a structure with reconverging fanout;

2) a circuit represented by one element, the vector-logical model of which is obtained using the method of Cartesian logic;

3) synthesized the Schneider circuit from the obtained logical vector by constructing conjunctive terms of the DNF from the unit coordinates of the vector. From the point of view of implementing proper functionality, all circuits are equivalent. From the point of view of implementing this circuit in a software or hardware product, the metric complexity of this action (algorithm) is determined by the 18-4-10 ratio. From the perspective of diagnostics, modeling, and testing these products, the complexity of their processing algorithms is determined by the 8-1-3 ratio. A 3-1-2 ratio determines the latency of each circuit in generating an output signal. The energy expenditure for the formation of the output result is 8-1-3. The conclusion is that the best implementation of the functionality is a logical vector placed in memory. Why, then, synthesize functionality into the logical elements of Emil Post's basis? The answer is simple – this is a tribute to the history of computing in the field of synthesis, which is studied by all universities worldwide. In addition, the student must be familiar with the fundamental concepts that enable the invention of new mechanisms for synthesis and analysis, which will be significantly more effective than the algorithms currently used to implement functionality. Modeling and synthesis can help each other by organizing harmonious relationships. Modeling involves organizing a smart model in memory, which can be enhanced in performance if this verified model is synthesized to implement functionality as a System-on-Chip (SoC) in high-speed quantum or matrix mechanisms, or ASIC chips. Modeling is a one-time procedure for creating a model of a process or phenomenon in memory, aimed at using it to model input binary sets to obtain a binary result representing a diagnosis or prognosis at the output of the circuit. Synthesis is a one-time procedure for implementing a verified functionality model into a system of crystal elements, as allowed by ASIC, VLSI, FPGA, CPU, or GPU.

Conclusion: Modeling and synthesis are in a harmonious relationship and help each other to implement cost-effective computing in terms of energy and time consumption.

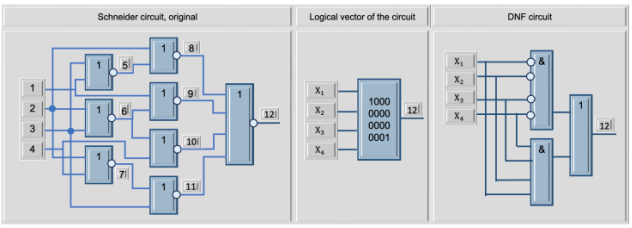

Fig. 8. **Three Schneider Circuit Models**
Source: **compiled by the authors**

4. HARMONY OF MODELING AND SYNTHESIS IN COMPUTING

Intelligent computing (Fig. 9) is a combination of three procedures: modeling (creating or synthesizing a model), simulation (forecasting or diagnosing on a developed or synthesized model), actuation (managing objects, processes or services on a created or synthesized model of a model), which generates four types of computing. The friendship between a computer and a human is a natural language interface for the user and a binary language interface for the computer. This is the trend of the fifth industrial revolution. Where there is a place for human creativity is in creating moral management mechanisms for computing, focused on saving energy and time while delivering high-quality cyber-socio-physical services. Summary – you need to feed the computer with binary data, vectors, and matrices. In response, we will develop an algorithmic framework for inference for the user or robot (autonomous agent). Intelligent computing design is the creation of a digitized binary model of a process or phenomenon to simulate any input situation to obtain inference in the form of control signals, prognosis, or diagnosis. Modeling is the energy consumption mechanism that is convenient for the verification and modeling of physical phenomena. Synthesis is a low-latency mechanism for modeling and managing autonomous agents in real time. A mistake in synthesis costs millions of dollars, while an error in modeling expenses amounts to hundreds of dollars. Therefore, the collaboration between modeling and synthesis is purely pragmatic, determined by the economics of computing. Modeling helps synthesis eliminate errors. In turn, synthesis can be helpful in modeling

the hardware implementation of algorithms for complex projects, thereby minimizing the time required for their verification. These are the harmonious relations between the mechanisms of synthesis and modeling.

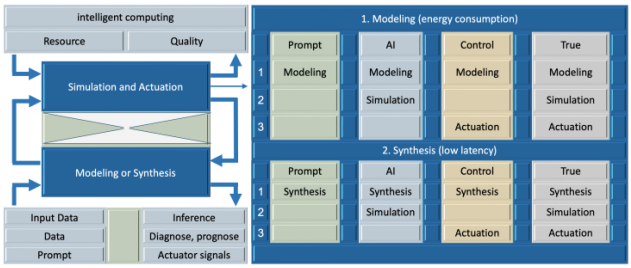

Fig. 9. **Harmony of modeling and synthesis in computing**

Source: **compiled by the authors**

CONCLUSION

The new mechanisms for modeling the logical vector of a digital circuit, which enables the creation of a map for testing the circuit's logical functionality are presented.

The mechanisms are free from the use of processor instructions and are focused on processing input data as logical vector addresses. At the same time, an in-memory computing architecture based on read-write transactions is employed, which enables energy savings of 36.3% and time savings of 22.6 % [29]. Additionally, the resulting logical vector of the circuit is the simplest and fastest parallel model for modeling good or faulty functionality. It is concluded that for modern high-speed memories (0.6 ns) [30], building a circuit vector is the most effective solution for implementing subsequent functionality in the SoC. Nobody needs the circuit today. Today, it has practically no advantages over memory in terms of speed and energy consumption. The circuit poses a challenge in solving verification and diagnosis problems related to defects. Functionality written to memory as a logical vector is free from synthesis and processor instructions. It is technologically advanced for modeling, testing, verification, and diagnostics. The logical vector is the basis of the new logical, economical mass computing (Fig. 10) for the maintenance of all existing spaces. The best data structures that a computer loves are the logical vector and its derivatives, truth tables, and matrices, which create a cost-effective and algorithmically efficient computing. The three components are all that is needed for binary processor-free in-memory modeling for simulation [13], [25], [26], [27] of any digitized process or phenomenon, represented as a vector, table, or matrix.

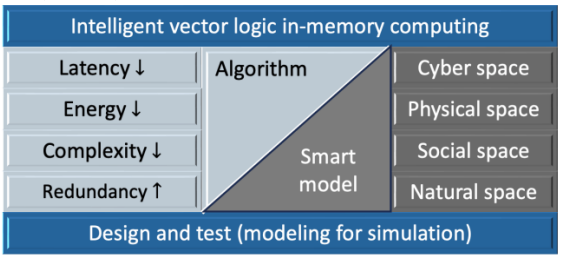

Fig. 10. **Map of vector-logical computing**

Source: **compiled by the authors**

The MOSI software application (Fig. 11) supports modeling the logical vector of the circuit and building a functionality test map with up to 16 input variables. The application GUI supports any logic presented in vector form. The number of circuit elements and their types is limited only by the user's imagination.

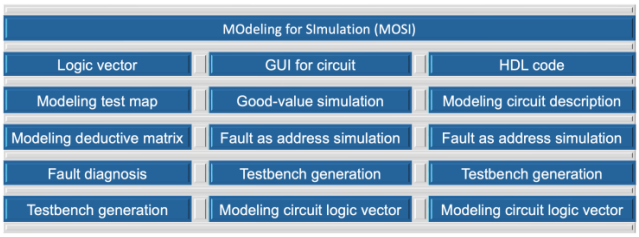

Fig. 11. **MOSI software application**

Source: **compiled by the authors**

The Modeling for Simulation software application is implemented in Python, contains approximately 10,000 lines of code, and can be installed on any computer or tablet. It is used as a cloud service or web application in the educational process for computer engineering students.

The authors of the study invite businessmen and scientists to capitalize on the vector-logical in-memory modeling for simulation in the EDA market.

REFERENCES

1. "IEEE Standard for Integrated Circuit (IC) Open Library Architecture (OLA)". In *IEEE STD 1481-2009*. 2010. p. 1–658. DOI: https://doi.org/10.1109/ieeestd.2009.5430852.

2. Esmaieli, E., Sedaghat, Y. & Peiravi, A. "Fanout-based reliability model for SER estimation in combinational circuits". In *IEEE Transactions on Circuits and Systems I: Regular Papers*. 2025; 72 (1): 228–240, https://www.scopus.com/pages/publications/85204697423.

DOI: https://doi.org/10.1109/tcsi.2024.3458864.

3. Hwang, M.-E., Jung, S.-O. & Roy, K. “Slope Interconnect Effort: Gate-Interconnect Interdependent Delay Modeling for Early CMOS Circuit Simulation”. In *IEEE Transactions on Circuits and Systems I: Regular Papers*. 2009; 56 (7): 1428–1441, https://www.scopus.com/pages/publications/67651156228. DOI: https://doi.org/10.1109/tcsi.2008.2006217.

4. Huang, W., Du, J., Hua, W., Bi, K. & Fan, Q. “A Hybrid Model-Based Diagnosis Approach for Open-Switch Faults in PMSM Drives”. In *IEEE Transactions on Power Electronics*. 2022; 37 (4): 3728–3732, https://www.scopus.com/pages/publications/85118596722.
DOI: https://doi.org/10.1109/tpel.2021.3123144.

5. Fu, R., Wille, R., Yoshikawa, N. & Ho, T.-Y. “Efficient Cartesian Genetic Programming-Based Automatic Synthesis Framework for Reversible Quantum-Flux-Parametron Logic Circuits”. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 2025; 44 (9): 3369–3380, https://www.scopus.com/pages/publications/85219316394. DOI: https://doi.org/10.1109/tcad.2025.3546884.

6. Berndt, A. et al. “Accuracy and Size Trade-off of a Cartesian Genetic Programming Flow for Logic Optimization”. *2021 34th SBC/SBMicro/IEEE/ACM Symposium on Integrated Circuits and Systems Design (SBCCI)*. Campinas, Brazil. 2021. p. 1–6, https://www.scopus.com/pages/publications/85116273904. DOI: https://doi.org/10.1109/sbcci53441.2021.9529968.

7. Dunaeva, O., Autsou, S., Kudelina, K. & Roosileht, M. “Utilizing the fuzzy logic algorithm for cartesian robot control system in conditions of mechanical damage transmission”. *IECON 2024 - 50th Annual Conference of the IEEE Industrial Electronics Society*. Chicago, IL, USA. 2024. p. 1–5, https://www.scopus.com/pages/publications/105000880837.
DOI: https://doi.org/10.1109/iecon55916.2024.10905445.

8. Khan, G. M., Zafari, F. & Mahmud, S. A. “Very short-term load forecasting using Cartesian Genetic Programming Evolved Recurrent Neural Networks (CGPRNN)”. *2013 12th International Conference on Machine Learning and Applications*. Miami, FL, USA. 2013. p. 152–155, https://www.scopus.com/pages/publications/84899412752. DOI: https://doi.org/10.1109/icmla.2013.181.

9. Inglese, P., Vatajelu, E.-I. & Di Natale, G. “Side channel and fault analyses on memristor-based logic in-memory”. In *IEEE Design & Test*. 2024; 41 (3): 29–35, https://www.scopus.com/pages/publications/85174850554.
DOI: https://doi.org/10.1109/mdat.2023.3324522.

10.Ye, J., Zhu, J., Cao, J., Bi, H., Ding, Y. & Chen, B. “A novel parallel in-memory logic array based on programmable diodes”. In *IEEE Journal of the Electron Devices Society*. 2024; 12: 738–744, https://www.scopus.com/pages/publications/85203846301. DOI: https://doi.org/10.1109/jeds.2024.3457021.

11.Chen, X., Song, T. & Han, Y. “RRAM-based analog in-memory computing: invited paper”. *2021 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*. AB, Canada. 2021. p. 1–6, https://www.scopus.com/pages/publications/85123977793.
DOI: https://doi.org/10.1109/nanoarch53687.2021.9642235.

12.Nguyen, V.-T., Trinh, Q.-K., Zhang, R. & Nakashima, Y. “XNOR-BSNN: In-memory computing model for deep binarized spiking neural network”. *International Conference on High Performance Big Data and Intelligent Systems (HPBD&IS)*, Macau, China, 2021, p. 17–21, https://www.scopus.com/pages/publications/85124890965.
DOI: https://doi.org/10.1109/hpbdis53214.2021.9658467.

13.Hahanov, V., Litvinova, E., Davitadze, Z., Chumachenko, S., Devadze, D. & Hacimahmud, A. V. “Truth table based intelligent computing”. *31st International Conference on Mixed Design of Integrated Circuits and System (MIXDES)*. Gdansk, Poland. 2024. p. 199–204, https://www.scopus.com/pages/publications/85201818702.
DOI: https://doi.org/10.23919/mixdes62605.2024.10614035.

14.Privman, V. & Solenov, D. “Decoherence of dynamically manipulated qubits”. *2006 Sixth IEEE Conference on Nanotechnology*. Cincinnati, OH, USA. 2006. p. 842–845.
DOI: https://doi.org/10.1109/nano.2006.1717240.

15.Durmuş, M. A., Demiralay, K., Khan, M. M., Atalay, Ş. E. & Sarpkaya, İ. “Moiré localization induced enhancement in the decoherence time of interlayer excitons in WSe2-MoSe2 Heterobilayers”. *Conference on Lasers and Electro-Optics (CLEO)*. Charlotte, NC, USA. 2024. p. 1–2. DOI: https://doi.org/10.1364/cleo_fs.2024.fw4b.2.

16.Zhu, S. et al. “Intelligent computing: the latest advances, challenges, and future”. *Intell Comput*. 2023; 2: 0006, https://www.scopus.com/pages/publications/85206596534. DOI: https://doi.org/10.34133/icomputing.0006.

17.Tarek, A., Alveed A. & Farhan, A. “A proof theoretic exploration of mathematical induction in computational paradigm”. *Congress in Computer Science, Computer Engineering, & Applied Computing (CSCE)*. Las Vegas, NV, USA. 2023. p. 963–972, https://www.scopus.com/pages/publications/85191183588. DOI: https://doi.org/10.1109/csce60160.2023.00162.

18.Kamide, N. “Natural deduction with explosion and excluded middle”. *IEEE 53rd International Symposium on Multiple-Valued Logic (ISMVL)*. Matsue, Japan. 2023. p. 24–29, https://www.scopus.com/pages/publications/85164603572.
DOI: https://doi.org/10.1109/ismvl57333.2023.00016.

19.Chang, S. -H., Liu, C.-N. J. & Küster, A. “Behavioral level simulation framework to support error-aware CNN training with in-memory computing”. *18th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*. Villasimius, Italy. 2022. p. 1–4, https://www.scopus.com/pages/publications/85134722352.
DOI: https://doi.org/10.1109/smacd55068.2022.9816307.

20.Li, S., Zhang, N., Zhang, W., Yang, R., Chang Y. & Xiong, B. “framework independent modeling for SRAM-based in-memory computing”. *2nd International Symposium of Electronic Design Automation (ISEDA)*. Xi'an, China. 2024. p. 777–777, https://www.scopus.com/pages/publications/85201735286. DOI: https://doi.org/10.1109/iseda62518.2024.10617799.

21.Westphal J. & Hardy, J. “Logic as a vector system”. In *Journal of Logic and Computation*. 2005; 15 (5): 751–765, https://www.scopus.com/pages/publications/26444498543.
DOI: https://doi.org/10.1093/logcom/exi040.

22.Mizraji, E. “Vector logic: a natural algebraic representation of the fundamental logical gates”. In *Journal of Logic and Computation*. 2008; 18 (1): 97–121, https://www.scopus.com/pages/publications/38749097766. DOI: https://doi.org/10.1093/logcom/exm057.

23.Mizraji, E. “Vector logic allows counterfactual virtualization by the square root of NOT”. In *Logic Journal of the IGPL*. 2020; 29 (5): 859–870, https://www.scopus.com/pages/publications/85110955658. DOI: https://doi.org/10.1093/jigpal/jzaa026.

24.Hahanov, V., Gharibi, W., Chumachenko, S. & Litvinova E. “Vector synthesis of fault testing map for logic”. *IAES International Journal of Robotics and Automation (IJRA)*. 2024; 13 (3): 293–306, https://www.scopus.com/pages/publications/85202981781. DOI: https://doi.org/10.11591/ijra.v13i3.pp293-306.

25.Hahanov, V., Litvinova, E., Hahanova, H., Chumachenko, S., Davitadze, Z., Hahanova, I., Kulak, H., Ponomarova, V. & Abdullayev V. H. “Vector-logical in-memory simulation of faults as truth table addresses”. *2024 IEEE East-West Design & Test Symposium (EWDTS)*. Yerevan, Armenia. 2024. p. 1–6, https://www.scopus.com/pages/publications/86000014942.
DOI: https://doi.org/10.1109/ewdts63723.2024.10873615.

26.Hahanov, V., Devadze, D., Hahanov, I., Chumachenko, S., Litvinova, E., Obrizan, V., Pashkov, D., Mishchenko, A. & Maksymova, N. “Prompt-testing of logic”. *IEEE East-West Design & Test Symposium (EWDTS)*. Yerevan, Armenia. 2024. p. 1–5, https://www.scopus.com/pages/publications/86000011281. DOI: https://doi.org/10.1109/ewdts63723.2024.10873774.

27.Hahanov, V., Chumachenko, S., Litvinova, E., Hahanov, I., Ponomarova, V., Khakhanova, H. & Kulak, G. “Faults-as-address simulation”. *IAES International Journal of Robotics and Automation*. 2024; 13 (4): 452–468, https://www.scopus.com/pages/publications/85212867387.
DOI: https://doi.org/10.11591/ijra.v13i4.pp452-468.

28.Ubar, R., Raik, J., Jenihhin, M. & Jutman, A. “Structural decision diagrams in digital test: theory and applications”. *Birkhäuser Cham, Computer Science Foundations and Applied Logic Series*. 2024. DOI: https://doi.org/10.1007/978-3-031-44734-1.

29.Ahn, B., Jang, J., Na, H., Seo, M., Son, H. & Song, Y. H. “AI accelerator embedded computational storage for large-scale DNN models”. *IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. Incheon, Korea Republic. 2022. p. 483–486, https://www.scopus.com/pages/publications/85139051812.
DOI: https://doi.org/10.1109/aicas54282.2022.9869991.

30.Wu, B., Zhu, H., Chen, K., Yan C. & Liu, W. “MLiM: High-Performance Magnetic Logic in-Memory Scheme with Unipolar Switching SOT-MRAM”. In *IEEE Transactions on Circuits and Systems I: Regular Papers*. 2023; 70 (6): 2412–2424, https://www.scopus.com/pages/publications/85151381814. DOI: https://doi.org/10.1109/tcsi.2023.3254607.

Conflicts of Interest: The authors declare that they have no conflict of interest regarding this study, including financial, personal, authorship or other, which could influence the research and its results presented in this article

Received 08.10.2025
Received after revision 28.11.2025
Accepted 03.12.2025

DOI: https://doi.org/10.15276/aait.08.2025.28
УДК 004.582

Векторні логічні структури та функції обчислення

Хаханов Володимир Іванович[1)]
ORCID: https://orcid.org/0000-0001-5312-5841; hahanov@icloud.com. Scopus Author ID: 7801667873
Чумаченко Світлана Вікторівна[1)]
ORCID: https://orcid.org/0000-0001-8913-1194; svetlana.chumachenko@nure.ua. Scopus Author ID: 57188710840
Литвинова Євгенія Іванівна[1)]
ORCID: https://orcid.org/0000-0002-9797-5271; eugenia.litvinova@nure.ua. Scopus Author ID: 25650378900
Хаханова Ганна Володимирівна[1)]
ORCID: https://orcid.org/0000-0002-4528-6861; anna.hahanova@nure.ua. Scopus Author ID: 8326375900

Обрізан Володимир Ігорович[1)]
ORCID: https://orcid.org/0000-0002-1835-4056; vladimir.obrizan@nure.ua. Scopus Author ID: 15127546800
Максимова Наталія Георгіївна[1)]
ORCID: https://orcid.org/0009-0006-0293-655X; nataliya.maksymova@nure.ua. Scopus Author ID: 59669839800
[1)] Харківський національний університет радіоелектроніки, пр. Науки, 14. Харків, 61166, Україна

АНОТАЦІЯ

Векторно-логічні обчислення є економічно ефективним механізмом інтелектуальних обчислень у пам’яті, що використовує транзакції читання-запису для розв’язання практичних задач аналізу та керування фізичними, соціальними й бізнес-процесами на основі моніторингу. Поняття механізму вводиться як гармонійна взаємодія між моделлю та алгоритмом обчислень. Дослідження спрямоване на суттєве зниження часових і енергетичних витрат, пов’язаних із процесами моделювання у фізичному, соціальному та цифровому світах.Засобом досягнення цієї мети є використання векторно-логічних механізмів обчислень у пам’яті, які істотно спрощують алгоритми завдяки експоненційній надлишковості структур даних. Розглянуто механізми зменшення обчислювальної складності алгоритмів, що зазвичай має експоненційний характер через застосування структур даних на основі векторної логіки. Проведено аналіз функцій і структур з погляду алгоритмічної простоти для задач моделювання та симуляції. Запропоновано кілька механізмів, побудованих на картезіанській логіці, для моделювання логічного вектора зі структури цифрової схеми. Булевий вектор використовується для створення тестової карти за допомогою трьох матричних операцій. Розроблено хмарний сервіс MOSI, який моделює поведінку та відмови цифрових схем і їхніх функціональностей, використовуючи адреси таблиць істинності.

Ключові слова: комп'ютерна гра; інфрачервона камера; розпізнавання шаблонів; база даних, просторовий опис суглобів людини; фізичні вправи

ABOUT THE AUTHORS

Vladimir Ivanovych Hahanov - Doctor of Engineering Sciences, Professor, IEEE Senior Member, Design Automation Department. Kharkiv National University of Radio Electronics. 14, Nauky Ave. Kharkiv, 61166, Ukraine
ORCID: https://orcid.org/0000-0001-5312-5841; hahanov@icloud.com. Scopus Author ID: 7801667873
Research field: Cybersecurity; Design Automation; Intelligent Systems

Хаханов Володимир Іванович - доктор технічних наук, професор, старший член IEEE. Кафедра автоматизації проектування. Харківський національний університет радіоелектроніки, пр. Науки, 14. Харків, 61166, Україна

Svetlana Viktorivna Chumachenko - Head of the Design Automation Department, Member of EMC, Member of STC, Executive Secretary of CEUR-WS IT Conference Chairs. Kharkiv National University of Radio Electronics. 14, Nauky Ave. Kharkiv, 61166, Ukraine
ORCID: https://orcid.org/0000-0001-8913-1194; svetlana.chumachenko@nure.ua. Scopus Author ID: 57188710840
Research field: Design Automation; Information Technologies; Intelligent Decision Support

Чумаченко Світлана Вікторівна - завідувач кафедри Автоматизації проектування, член EMC, член НТК, відповідальний секретар CEUR-WS IT Conference Chairs. Харківський національний університет радіоелектроніки, пр. Науки, 14. Харків, 61166, Україна

Eugenia Ivanivna Litvinova - Professor, Scientific Secretary of the Specialized Scientific Council, Design Automation Department. Kharkiv National University of Radio Electronics. 14, Nauky Ave. Kharkiv, 61166, Ukraine
ORCID: https://orcid.org/0000-0002-9797-5271; eugenia.litvinova@nure.ua. Scopus Author ID: 25650378900
Research field: Automated Design Systems; Information Systems; Intelligent Modeling

Литвінова Євгенія Іванівна - професор, учений секретар спеціалізованої вченої ради, кафедра Автоматизації проектування. Харківський національний університет радіоелектроніки, пр. Науки, 14. Харків, 61166, Україна

Hanna Volodymyrivna Hahanova - Doctor of Engineering Sciences, Professor, 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: Design Automation; Computational Intelligence; Software Engineering

Хаханова Ганна Володимирівна - доктор технічних наук, професор кафедри Автоматизації проектування. Харківський національний університет радіоелектроніки, пр. Науки, 14. Харків, 61166, Україна

Volodymyr Ihorovych Obrizan - PhD in Computer Engineering, Member of IEEE, Senior Lecturer, Design Automation Department. Kharkiv National University of Radio Electronics. 14, Nauky Ave. Kharkiv, 61166, Ukraine
ORCID: https://orcid.org/0000-0002-1835-4056; vladimir.obrizan@nure.ua. Scopus Author ID: 15127546800
Research field: Computer Engineering; Automation of Design; Software Reliability

Обрізан Володимир Ігорович - кандидат технічних наук, старший викладач кафедри Автоматизації проектування. Харківський національний університет радіоелектроніки, пр. Науки, 14. Харків, 61166, Україна

Nataliya Heorhievna Maksymova - IT Consultant | IT Project Manager | Lecturer, Design Automation Department. Kharkiv National University of Radio Electronics. 14, Nauky Ave. Kharkiv, 61166, Ukraine
ORCID: https://orcid.org/0009-0006-0293-655X; nataliya.maksymova@nure.ua. Scopus Author ID: 59669839800
Research field: IT Project Management; Information Systems; Digital Transformation

Максимова Наталія Георгіївна - IT-консультант, керівник IT-проєктів, викладач кафедри Автоматизації проектування. Харківський національний університет радіоелектроніки, пр. Науки, 14. Харків, 61166, Україна