Date of publication SEP-30, 2023, date of current version JUL-05, 2023.
www.computingonline.net / computing@computingonline.net

Print ISSN 1727-6209
Online ISSN 2312-5381
DOI 10.47839/ijc.22.3.3227

Vector-deductive Faults-as-Address Simulation

Anna Hahanova
Kharkov National University of Radio Electronics, Kharkiv, 61166, Ukraine

Corresponding author: Anna Hahanova (e-mail: hahanov@icloud.com).

ABSTRACT The main idea is to create logic-free vector simulation, based on only read-write transactions on address memory. Stuck-at fault vector simulation is leveraged as a technology for assessing the quality of tests for complex IP-cores implemented in Field Programmable Gate Array (FPGA), Application-Specific Integrated Circuit (ASIC). The main task is to implement new simple and reliable models and methods of vector computing based on primitive read-write transactions in the technology of vector flexible interpretive fault simulation. Vector computing is a computational process based on read-write transactions on bits of a binary vector of functionality, where the input data is the addresses of the bits. A vector-deductive method for the synthesis of vectors for propagating input fault lists is proposed, which has a quadratic computational complexity. Analytical expressions of logic that require algorithmically complex computing are replaced by vectors of output states of elements and digital circuits. A new matrix of deductive vectors is synthesized, which is characterized by the following properties: compactness, parallel data processing based on a single read–write transaction in memory, exclusion of traditional logic from fault simulation procedures, full automation of its synthesis process, and focus on the technological solving of many technical diagnostics problems. A new structure of the sequencer for vector deductive fault simulation is proposed, which is characterized by ease of implementation on a single memory block. It eliminates any traditional logic, uses data read-write transactions in memory to form an output fault vector, uses data as addresses to process the data itself.

KEYWORDS vector computing; vector form of logic; matrix of deductive vectors; deductive-vector fault-as-address simulation; read-write transaction; vector model of input faults; functions and structures; sequencer of vector deductive fault simulation.

I. INTRODUCTION

The motivation for the research is defined by the following factors: the use of elementary read-write transactions in memory-driven computing based on a vector description of logic, memory redundancy for storing interpretive flexible logic models, the use of data as addresses to increase the speed of deductive simulation, transfer of computing to a lower level of computational processes (read-write transaction), where the von Neumann architecture and the Post-Jablonski theorem about functional completeness can be ignored [1, 2]. The essence of vector computing is read-write transactions on vector data structures in address memory. The relevance of this direction can be seen from the latest Gartner Hype Cycle 2022, which highlighted Computational Storage (CS) as a trigger trend of transferring data processing from the CPU to the memory where they are located [3]. Big data must be processed where it is stored.

One of these solutions is vector computing that is a computational process based on read-write transactions on the bits of a binary vector of functionality that forms Computational Storage, where the input data (Conventional Memory) are bit addresses. Data (fault vectors) in the proposed vector-deductive simulation method are used as addresses for processing the data itself.

The input data models can be represented (Fig. 1) by follows: 1) sets – they are compact data that require a complex and sequential algorithm for processing inputs. 2) vectors, which provide unitary data coding, use a parallel register algorithm for data processing and sequential algorithm for processing inputs. 3) addresses, which provide a compact encoding of unitary data and a sequential algorithm for their processing by read-write transactions in memory free of logic and processor with parallelism in address columns.

In design and test, three main forms of describing processes and phenomena are used: tabular and analytical forms, and graphs [4-9]. In this case, the matrix (table) and the vector are two forms of describing models that pass into each other. A vector (binary, multi-valued) is a compact representation of a truth table in the form of an ordered sequence of output states, if the input address components are sorted in ascending order [10-13]. If necessary, the matrix, can be transformed into a one-dimensional vector for the convenience of parallel data processing in register memory. Naturally, it is enough to simply

restore the table or matrix from the vector description form of the process or phenomenon.

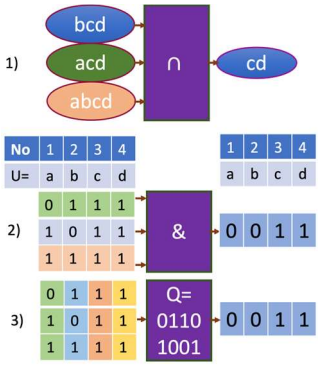

Figure 1. Input data models

Further, a vector is used as the simplest form of describing functions and structures [14-19], which is focused on placement in address memory (CS).

Several approaches were proposed [20-23] to testing (Built-in self-test, BIST), diagnosing and repairing the memory structures based on redundancy, as well as to constructing and evaluating redundancy analysis algorithms based on preference vectors for memory devices with spare elements. Experiments on the use of new algorithms for self-testing and repair (STAR) of SRAM memory have shown the effectiveness of the proposed approach. There is a trend: all design and test, diagnosis and repair problems are technologically solved more simply on regular memory for processing big data, than on a classic computer that uses a standard ALU-based processor. The bottleneck of a classical computer is the communication channel between a slow memory and a fast logical processor. At present, memory performance has increased tenfold, so the communication channel with the logical processor is becoming an unacceptable obstacle to improving the performance of computer architecture. The solution is to transfer all calculations to memory, remove the logical processor from the computer architecture, at the same time solving the problem of bottlenecks in the interaction between memory and the logical processor (Y. Zorian, Yerevan, 2007 IEEE EWDTS).

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

Efficient fault simulation algorithms [24–26] for combinational circuits have been known for several decades. However, sequential failure simulation, which is often used in testing and fault tolerance applications, remains a very time-consuming problem, especially for large circuits. A new deductive method is proposed for simulating errors at the RTL level and in the system model of high-level decision diagrams. Simulation acceleration is achieved through efficient data structures implemented to perform a set of operations in the deductive fault simulation algorithm. Experiments on RTL reference circuits show that this method achieves up to two orders of magnitude reduction in operating time compared to failure simulation at the gate level. Raimund Ubar singles out deductive simulation as the most elegant mathematical apparatus, which has good prospects for processing complex digital circuits.

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

General conclusions are represented below. The crisis of modern computing is associated with two problems: processing big data by an insolvent processor-memory pair, as well as a catastrophic increase in power consumption by global computing processes on modern microelectronics. Solving the first problem is based on the following axioms: All data is in memory. There is no logic that cannot be implemented in memory. There is no data that cannot be used as addresses to be processed in the memory, where the logic resides. There is no logic or functionality that cannot be implemented with a read-write transaction on memory. The most technologically advanced memory data structure is a vector or matrix, available for fast read-write transactions. Solving the second problem is based on hypotheses: quantum computing should be based on memory free of quantum logic and qubits. Memory must be stable subatomic particles. Computational processes should be based on quantum transactions between memory elements. The source of energy for such a computer is daylight. The speed of such a computer is determined by the light speed of quantum transactions between memory elements. Such a hypothetical computer is the meeting point of quantum (seeking determinism) and classical (seeking light speed) computing in the future.

The goal is to develop a vector-deductive fault simulation method based on primitive read–write transactions for analyzing logic circuits. Objectives: 1) Development of a method for synthesizing a matrix of deductive vectors for propagating input fault vectors to the element output. 2) Development of a sequencer structure for simulating faults of digital circuits based on a primitive read-write transaction of matrix memory, where combinations of faults are addresses. 3) Verification and testing of models and methods for synthesizing matrices of deductive vectors and a sequencer for deductive fault analysis in digital elements and circuits.

II. DEDUCTIVE VECTOR MATRIX SYNTHESIS METHOD FOR FAULT SIMULATION

Induction is drawing logical conclusions from the particular to the general, where the correctness is guaranteed by a sufficient or exhaustive amount of factual data. It is mother of artificial intelligence. Deduction is drawing logical conclusions from the general to the specific, where the correctness is guaranteed by the truth of the initial axioms leading to the truth of the consequences-theorems. It is the mother of deterministic computing. The mathematical basis of deductive fault simulation consists in propagating binary combinations of input defects to the output on a given input set according to the following formula: $L = T \oplus F$. The essence of deductive simulation is to change the logic of the element F depending on the input conditions T. Deductive simulation, proposed exactly 50 years ago by Armstrong [27], is still the most elegant and effective tool for analyzing the quality of tests and synthesizing tables for detecting faults, tracing the path propagation of the fault. Further, its implementation is proposed based on the vector form of the logic description [10–12], which excludes logical analytical forms and makes it possible to significantly

simplify the algorithms for the synthesis of deductive models and their application for interpretative modeling digital elements and large-scale circuits.

The goal of this research is to remove all analytical expressions of deductive logic that appeared in most works [4-9] devoted to deductive simulation. It is possible and necessary to use only the deductive vector of fault simulation. Very often the form of the model determines the content, compactness, and speed of the analysis procedures. The basic methods of deductive simulation [5-14] were reduced to the synthesis of deductive analytic forms (DNF). Information about the logical element was represented by a truth table or a logical expression. But in any case, explicitly or implicitly, the equation of technical diagnostics L=T$\oplus F$ was used to obtain deductive forms of fault simulation. Further, a new method for the synthesis of deductive vectors is proposed, which is reduced to the analysis of output state vector of a logical element free of a truth table.

The method for synthesizing deductive vectors using a vector form of any logic functionality (Q-vector) has the following two steps:

1) Modification of the Q-vector of the element (Fig. 3) on the input i-set according to the rule: L=Q$\oplus Y_i$. Here Y_i is the state of the logical element output on the input set x_i.

2) Determination of the deductive vector for the i-input set by the formula: $D_j = L_{H_{ij}}, j = \overline{1, 2^n}$, which permutes the bits according to the following permutation matrix H, which can be easily obtained through recursion:

$$H_{ij}(n=1,2,3) = \begin{matrix} 0 & 1 \\ 1 & 0 \end{matrix} \rightarrow \begin{matrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 3 & 2 \\ 2 & 3 & 0 & 1 \\ 3 & 2 & 1 & 0 \end{matrix} \rightarrow \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 0 & 3 & 2 & 5 & 4 & 7 & 6 \\ 2 & 3 & 0 & 1 & 6 & 7 & 4 & 5 \\ 3 & 2 & 1 & 0 & 7 & 6 & 5 & 4 \\ 4 & 5 & 6 & 7 & 0 & 1 & 2 & 3 \\ 5 & 4 & 7 & 6 & 1 & 0 & 3 & 2 \\ 6 & 7 & 4 & 5 & 2 & 3 & 0 & 1 \\ 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \end{matrix}$$

The algorithm ends when all deductive vectors for all input 2^n sets have been generated. Thus, the matrix of deductive vectors is obtained based on the execution of the following operator D=$(Q \oplus Y)_H$ that is a result of the superposition of the operators: L=Q$\oplus$Y and D=L_H. The synthesis of matrices of deductive vectors for 2-input based logic is shown in Fig. 2.

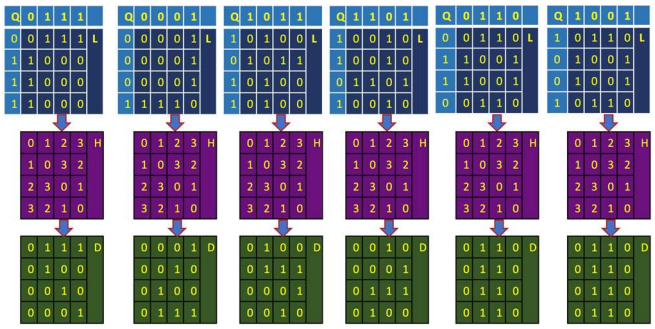

Figure 2. Synthesis of deductive vectors for 2-input based logic

It should be noted that the synthesis of deductive formulas for mutually inverse elements Q=0110 and Q=1001 gives the same values of the matrix of deductive vectors, which degenerate into one vector 0110 on all input sets. The computational complexity of executing this operator is C= $2\times 2^n \times 2^n = 2^{2n+1}$. In the case of parallel execution of register operations on vectors, the computational complexity of this operator will be equal to C= 2×2^n.

III. SYNTHESIS OF DEDUCTIVE VECTORS OF GATE AND RTL LOGIC

The process of building deductive models of the main logic elements is also of great interest, which can be used as a library for creating and analyzing circuits. Below are tables for the synthesis of deductive vectors to check the quality of tests of logical circuits. The most primitive elements are the inverter (Q=10) and the repeater (Q=01). Even though these are different elements, they have the same deductive vectors, which allow providing digital logical activity to transport the fault vector from input to output without distorting it.

The next Frame is devoted to the process of synthesizing deductive formulas for a three-input logic element given by the vector coverage 10000001 (Fig. 3). Such an element should be considered as a black box or Register Transfer Level (RTL) of function representation in relation to its structure, which can be implemented differently when specifying its behavior in a vector. Here, the result of propagating lists of activities from the input to the output of this element is of interest. In this case, it is not important which paths are involved within a particular implementation of a logical element. Nevertheless, the synthesis of deductive formulas for this element showed that on all input actions the activity propagation formula has the same value on pairs of sets: 1–16 and 5–6. The remaining sets, having symmetry, are not repeated in the matrix of deductive vectors. Here and below, the zero coordinates of the matrices L and D are represented by empty cells for the purpose of figurative perception of information.

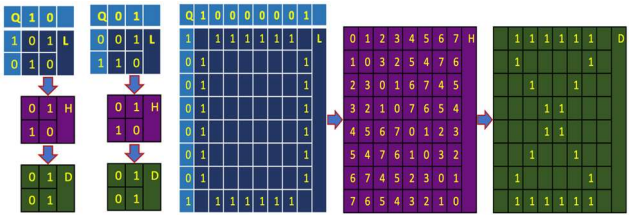

Figure 3. Synthesis of deductive vectors for one-input 10 and 01 and three-input 10000001–elements

The following circuit (Fig. 4) has the property that the deductive vectors in the generated matrix of deductive vectors MDV are the same and equal to 00110011. The analytic form of such a vector after elementary transformations is D=X_2 on all input sets. This means that two of the three functionality variables are non-essential and cannot be activated by input faults.

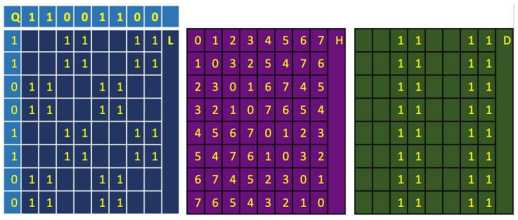

Figure 4. Synthesis of deductive vectors for a three-input 11001100–element

Increasing the number of inputs of RTL functional element leads to an increase in simulation performance since more input fault vectors are simulated in parallel to obtain an output fault

vector. The use of 8-input element improves simulation performance by a factor of 11 compared to its structural gate equivalent. In general case, the improvement in simulation performance for RTL circuits having n inputs, compared with the analysis of the DNF structure of two-input gates, is given by the following formula: $Q = \frac{n}{2} + n-1$, n= 4, 8, 16 …

The technological simplicity of the algorithm for synthesizing the H-matrix of coordinate permutation $H^i = \begin{bmatrix} H_1^{i-1} & H_2^i \\ H_3^i & H_4^{i-1} \end{bmatrix}$ is of interest, i=1,2,3… is number of input variables, which has 3 items:

1) the first and fourth quarters of the matrix are taken from the previous recursive calculation of the matrix for *n*=i–1 variable, here the following equalities are satisfied: $H_1^{i-1} = H_4^{i-1}, H_2^{i-1} = H_3^{i-1}$;

2) the second quarter of the matrix is found based on the following expression (Fig. 5):

$H_{i,j+2^{n-1}} = (2^n - 1) - H_{i,2^{n-1}-j}, j = 0,2^{n-1}-1, i = 0,2^{n-1}-1$;

3) the third quarter of the matrix is found by copying the second part of the matrix into the third area $H_3^i = H_2^i$.

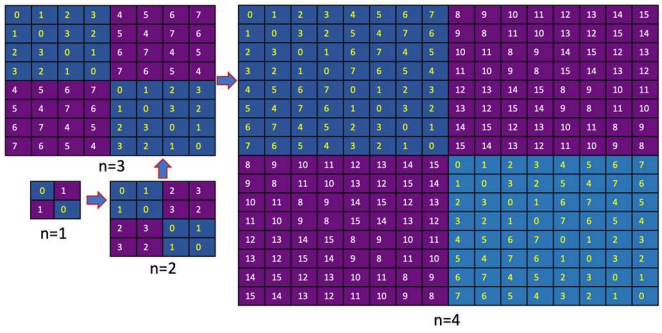

Figure 5. Synthesis of vector recoding matrix D=L(H)

Despite the technological simplicity of the proposed method for synthesizing a matrix of deductive vectors, this approach has an obvious drawback associated with the dimension of the tables with many input variables. It can be eliminated if only one deductive vector is promptly generated on the input test set to simulate faults. To do this, you just need to use a single operator $D_i=(Q \oplus Y_i)_{H_{ij}}$, described earlier. The computational complexity of this procedure is equal to 2^n, n is the number of variables in the logical element, which is determined by permuting the bits in the Q-vector of H-matrix to obtain the D-vector. In this case, the deductive simulation of faults will not differ much in speed from the fault-free simulation of a digital circuit. The processing time delta of one logic element $\Delta T=TD-TG=tD+tF=2^{n+1}$ will be represented by the deductive vector generation time tD and plus the processing time of input fault vectors tF.

IV. VECTOR–DEDUCTIVE SEQUENCER

The matrix of deductive vectors is a certain redundancy of a digital project, which is the cost for a fast and technological solution to the problem of assessing the quality of test patterns and generating a fault functions table to detect faults at the stage of a digital product functioning. The proposed matrix of deductive vectors has the following properties: 1) compactness; 2) parallel data processing; 3) technological placement in address memory; 4) data uniformity in size and properties; 5) simultaneous simulation of fault-free behavior of the element and all faults of previous elements; 6) focus on the technological solution to the problems of simulation, testing, and diagnostics of any logical systems.

The vector sequencer of deductive simulation on a memory block (Fig. 6) is the simplest implementation of computing device for deductive fault simulation of a digital circuit (element) to assess the quality of the tests in the class of single stuck-at faults. The main and single memory block stores a matrix of deductive vectors, which has a dimension of $2^n \times 2^n$, where the first number is the number of deductive vectors for a logical element of n-variables, the second one is the dimension of each vector. Therefore, the first macro input of the memory block Vector address has n binary variables that allow addressing any of the 2^n deductive vectors of the matrix. The second macro input of the memory block Byte address also has n, but they are already register binary variables, which allow addressing any of the 2^n bits of the vector, selected by the first macro input, by their binary combination. The output of the memory block Fault list has a bit width equal to the second macro input, determined by the power of the simulated input faults that must be propagated through the element of the digital structure.

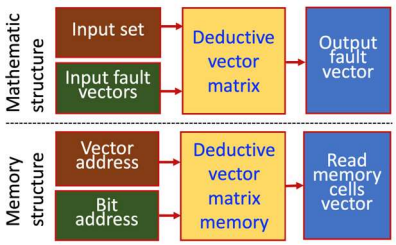

Figure 6. Vector sequencer of deductive simulation implemented in the memory block

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

The metric of a functional n-input element F, represented by a vector of 2^n states of output coordinates and its deductive model DF, represented by MDV matrix of 2^n deductive vectors with a dimension of 2^n binary coordinates, is shown in Fig. 7.

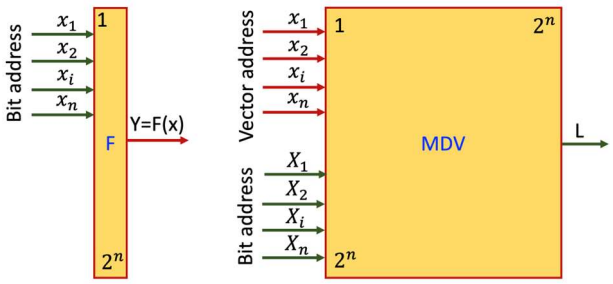

Figure 7. Metrics of functional and deductive elements

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

Thus, the ratio of the two models by memory is as follows: $F/DF=2^n/(2^n \times 2^n)=1/2^n$. The computational complexity of synthesizing a matrix of deductive vectors is estimated by the following expression: $Q=n\times 2^n + 2^n \times 2^n$, the first term determines the complexity of generating address-disordered input variables of the deductive vectors in register operations on vectors, the second term determines the complexity of the coordinate operations for reducing the bits of deductive vectors according to the order of the binary addresses, composed by input variables. The computational complexity of the analysis of the matrix of deductive vectors when performing deductive simulation is $Q=k\times R$, where k is the dimension of the input fault vectors, R is the duration of the operation of reading the contents of the addressable bit of the deductive vector from the MDV memory. The significant deductive redundancy of the project pays for the quality and reliability of the digital system, which also allows generating tests, evaluating their quality, and solving any problems related to diagnosing faults in the design and operation of a digital device in critical areas of human activity.

V. VERIFICATION OF THE VECTOR-DEDUCTIVE METHOD FOR FAULT SIMULATION

An example of vector-deductive simulation of the circuit C17 from ISCAS library with reconvergent fan-outs (2,7,8) on one input set 11111 is represented by a table (Fig. 8). In this case, a class of single stuck-at faults is considered, which are tied to input, internal or output lines (of the same potential) of the circuit.

Any line fault that is detected by this test is marked by one in the coordinate of the fault simulation table. The actual fault of the line will always be the inverse of its fault-free state. Vectors are also used to simulate the fault-free behavior of the circuit. This circuit uses one vector Q=1110, which is used to simulate all elements of the circuit. The procedure of vector interpretive memory-driven fault-as address simulation is described in [15, 17, 19]. In publication [20] the authors considered faults as sets or vectors.

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

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

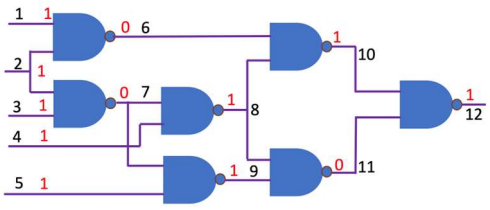

Figure 8. Vector–deductive simulation of C17 on the input set 11111

The right side of the table contains a matrix of deductive vectors for the convenience of manual simulation. The input word x_1x_2 on the circuit element (00,01,10,11) forms the address of the deductive vector, the coordinates of which are chosen by a pair of values of the input fault vectors to generate the output list of detected faults. The preparation of the fault simulation table consists in setting unit values along the diagonal of the matrix. Empty matrix coordinates denote zeros. The simulation ends when all fault vectors are sequentially generated for all 12 lines of the circuit. In fact, this means processing all seven circuit elements. The processing of the element with output 12 for generating the output fault list is shown in more detail in Fig. 9.

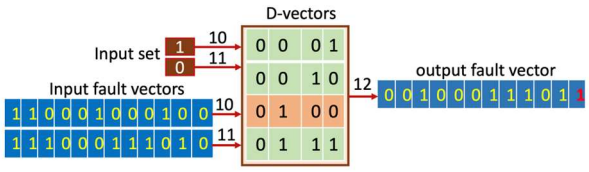

Figure 9. Vector–deductive simulation of an element with an output of 12 on an input set of 10

Here, the logical values of the inputs (10,11) = 10 select the deductive simulation vector 0100. By successively reading the coordinates of this vector, an output fault list is formed based on the addresses generated by the corresponding pairs of coordinates (11→ 0, 11→ 0, 01→ 1, 00→ 0, 00→ 0, 10→ 0, 01→ 1, 01→ 1, 01→ 1, 10→ 0, 01→ 1, 00→ 0) of two input fault lists. The unit at the output of the processed element marked in red in the table is the dominant value at the output of the simulated component. Otherwise, a fault at the output of the simulated element is always detected. Therefore, the diagonal coordinate of the matrix with the number $M_{ij,}i = j$ does not need to be simulated as an input fault. The computational complexity of the proposed vector-deductive method is determined by the parameters of the following metric:

$Q=\frac{1}{2}\times k\times n^2$, where k is the time to read a vector bit from memory; n is the number of lines in the circuit; 1/2 is half of the simulation table. Another simulation example is shown in Fig. 10.

	1	2	3	4	5	6	7	8	9	10	11	12
States	1	1	0	0	1	0	1	1	0	1	1	0
1	1											
2		1								x_1x_2	Q=1110	
3			1							00	0	0 0 1
4				1						01	0	0 1 0
5					1					10	0	1 0 0
6	1	1				1				11	0	1 1 1
7		0	1				1					
8		0	0	1			0	1				
9		1	1	1			1		1			
10	1	1	0	0		1		0		1		
11	0	1	1	0			1	0	1		1	
12	1	1	1	0		1	1	0	1	1	1	1
Faults	0	0	1			1	0		1	0	0	1

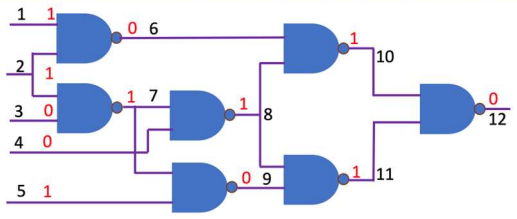

Figure 10. Vector–deductive simulation of a circuit on an input set 11001.

The test set 11001 0110110 detects 9/24=37.5% of the faults, which are determined by inversion to the states vector (the fault-free state) of the circuit lines, which is needed only to decoding single stuck-at faults detected on the circuit lines. Based on the results of simulating input binary vectors, a matrix of fault coverage by test sets is generated, which for two simulated vectors has the following form (Fig. 11):

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

Figure 11. Fault coverage matrix

The symbol x = {0,1} denotes a detection on the line ≡0 and ≡1. The test is considered complete if all coordinates in the coverage line are equal to the x symbol. The method uses vector data structures, which makes it interesting from the point of view of its implementation in computer memory or built-in specialized hardware device. The deductive simulator is free of traditional logic and uses the coordinates of the input fault vectors as bit addresses of the deductive vectors that form the output fault list to be detected. The simulator is easily implemented in any memory, including at the quantum level, where only one read-write transaction is required. The paradigm of interpreting data as addresses can be promising when processing large data using read-write transactions on memory free of traditional logic and powerful expensive processors. The idea could create a new type of deterministic quantum computing free of qubits and quantum logic, based on photonic transactions on a stable quantum atomic memory.

VI. CONCLUSIONS

A certain step has been taken towards the creation of logic-free vector memory-based computing, using only read-write transactions on address memory. A failure-driven management metric $T \oplus F \oplus L = 0$, proposed in [18, 20], allows formalizing all known processes for creating computing, including design, test, and deductive fault simulation.

Based on the metric, innovative solutions are proposed below. The advantages of the vector model for a compact description of processes, phenomena, functions and structures are determined. Analytical formulas and tabular models that require algorithmically complex computing analyzers are replaced by vector data structures for describing functional and deductive logic.

A vector-deductive method for the synthesis of vectors (instead of formulas) for propagating input fault vectors is proposed, which has a quadratic computational complexity of register operations. A new matrix of deductive vectors is proposed, which is characterized by the following properties: compactness, interpretability and flexibility of models written to memory, parallel data processing based on a single read-write transaction in memory, technological placement in address memory, uniformity of data in size and properties, exclusion of fault-free simulation of the element, exclusion of traditional logic from fault simulation procedures, complete automation of the process of deductive vector matrix synthesis, focus on the technological solving all problems of technical diagnostics.

A new structure of the vector deductive fault simulation sequencer is proposed, which is characterized by ease of implementation on a single memory block, free of any traditional logic, uses data read-write transactions in memory to form the output fault vector, uses data as addresses to process the data itself. The method can be used for parallel processing big data, which is interpreted as cell addresses of deductive and/or logical vectors that compose the computational memory on which read-write transactions but not logic ones are performed. The method can be the basis for a new deterministic quantum logic-free computing based on the execution of photonic (quantum) transactions on a structure of stable subatomic particles considered as memory. In addition, the proposed method can effectively solve the problem of recognizing any activity in the cyber-physical space.

The implementation of vector deductive logic models in the FPGA LUT will allow one to obtain the performance of fault simulation of real SoC digital blocks at the level of hundreds of nanoseconds [28, 29].

References

[1] C. E. Shannon, "Von Neumann's contributions to automata theory," *Bulletin American Mathematical Society*, vol. 64, 1958, in *Claude E. Shannon: Collected Papers*, IEEE, pp.831-835, 1993. https://doi.org/10.1090/S0002-9904-1958-10214-1.

[2] M. Davis, "Emil Post's contributions to computer science," *Proceedings of the Fourth Annual Symposium on Logic in Computer Science*, 1989, pp. 134-136.

[3] What's New in the 2022 Gartner Hype Cycle for Emerging Technologies, August 10, 2022.

[4] What's New in the 2022 Gartner Hype Cycle for Emerging Technologies. Aug. 10, 2022. [Online]. Available at: https://www.gartner.com/en/articles/what-s-new-in-the-2022-gartner-hype-cycle-for-emerging-technologies

[5] M. Abramovici, *Digital System Testing and Testable Design*, Comp. Sc. Press, 1998.

[6] N. Takahashi, N. Ishiura and S. Yajima, "Fault simulation for multiple faults by Boolean function manipulation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 4, pp. 531-535, 1994. https://doi.org/10.1109/43.275363.

[7] M. Srivastava, S. K. Goyal, A. Saraswat and G. Gangil, "Simulation models for different power system faults," *Proceedings of the 2020 IEEE International Conference on Advances and Developments in Electrical and Electronics Engineering (ICADEE)*, pp. 1-6, 2020, https://doi.org/10.1109/ICADEE51157.2020.9368915.

[8] Menon and Chappell, "Deductive fault simulation with functional blocks," *IEEE Transactions on Computers*, vol. C-27, no. 8, pp. 689-695, 1978, https://doi.org/10.1109/TC.1978.1675175.

[9] I. Pomeranz and S. M. Reddy, "Forward-looking fault simulation for improved static compaction," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 10, pp. 1262-1265, 2001, https://doi.org/10.1109/43.952743.

[10] Z. Navabi, *Digital System Test and Testable Design. Using HDL Models and Architectures*, Springer, 2011. https://doi.org/10.1007/978-1-4419-7548-5.

[11] I. Pomeranz and S. M. Reddy, "Hazard-Based Detection Conditions for Improved Transition Fault Coverage of Functional Test Sequences," *2009 24th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, Chicago, IL, USA, 2009, pp. 358-366, https://doi.org/10.1109/DFT.2009.11.

[12] M. Zolfy, S. Mirkhani and Z. Navabi, "Adaptation of an event-driven simulation environment to sequentially propagated concurrent fault simulation," *Proceedings Design, Automation and Test in Europe. Conference and Exhibition 2001*, Munich, Germany, 2001, pp. 823, https://doi.org/10.1109/DATE.2001.915173.

[13] S. Moazzeni, A. Emami and S. Poormozaffari, "An Optimized Simulation-Based Fault Injection and Test Vector Generation Using VHDL to Calculate Fault Coverage," *2009 10th International Workshop on Microprocessor Test and Verification*, Austin, TX, USA, 2009, pp. 55-60, https://doi.org/10.1109/MTV.2009.22.

[14] I. Pomeranz and S. M. Reddy, "Unspecified Transition Faults: A Transition Fault Model for At-Speed Fault Simulation and Test Generation," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 1, pp. 137-146, Jan. 2008, https://doi.org/10.1109/TCAD.2007.907000.

[15] L. Wu, M. K. Hussain, S. Abughannam, W. Müller, C. Scheytt and W. Ecker, "Analog fault simulation automation at schematic level with random sampling techniques," *2018 13th International Conference on Design & Technology of Integrated Systems In Nanoscale Era (DTIS)*, Taormina, Italy, 2018, pp. 1-4, https://doi.org/10.1109/DTIS.2018.8368549.

[16] V. I. Hahanov, S. M. Hyduke, W. Gharibi, E. I. Litvinova, S. V. Chumachenko and I. V. Hahanova, "Quantum models and method for analysis and testing computing systems," *Proceedings of the 2014 11th International Conference on Information Technology: New Generations*, Las Vegas, NV, 2014, pp. 430-434, https://doi.org/10.1109/ITNG.2014.125.

[17] M. Karavay, V. Hahanov, E. Litvinova, H. Khakhanova and I. Hahanova, "Qubit fault detection in SoC logic," *Proceedings of the 2019 IEEE East-West Design & Test Symposium (EWDTS)*, Batumi, Georgia, 2019, pp. 1-7, https://doi.org/10.1109/EWDTS.2019.8884475.

[18] V. Hahanov, *Cyber Physical Computing for IoT-driven Services*, New York, Springer 2018. https://doi.org/10.1007/978-3-319-54825-8.

[19] V. Hahanov, I. Yemelyanov, V. Obrizan and I. Hahanov, "'Quantum' diagnosis and simulation of SoC," *Proceedings of the 2015 XI International Conference on Perspective Technologies and Methods in MEMS Design (MEMSTECH)*, 2015, pp. 58-60.

[20] V. Hahanov, E. Litvinova, O. Shevchenko, S. Chumachenko, H. Khakhanova and I. Hahanov, "Vector models for modeling logic based on XOR-relations," *Proceedings of the 2022 IEEE 16th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)*, 2022, pp. 823-828, https://doi.org/10.1109/TCSET55632.2022.9766894.

[21] G. Harutunvan, V. A. Vardanian and Y. Zorian, "Minimal march tests for unlinked static faults in random access memories," *Proceedings of the 23rd IEEE VLSI Test Symposium (VTS'05)*, 2005, pp. 53-59, https://doi.org/10.1109/VTS.2005.56.

[22] M. Psarakis, D. Gizopoulos, A. Paschalis and Y. Zorian, "Sequential fault modeling and test pattern generation for CMOS iterative logic arrays," *IEEE Transactions on Computers*, vol. 49, no. 10, pp. 1083-1099, 2000, https://doi.org/10.1109/12.888044.

[23] M. Renovell, J. M. Portal, J. Figueras and Y. Zorian, "RAM-based FPGAs: a test approach for the logic," *Proceedings Design, Automation and Test in Europe*, Paris, France, 1998, pp. 82-88, https://doi.org/10.1109/DATE.1998.655840.

[24] M. Psarakis, D. Gizopoulos, A. Paschalis and Y. Zorian, "Sequential fault modeling and test pattern generation for CMOS iterative logic arrays," *IEEE Transactions on Computers*, vol. 49, no. 10, pp. 1083-1099, 2000, https://doi.org/10.1109/12.888044.

[25] U. Reinsalu, J. Raik, R. Ubar and P. Ellervee, "Fast RTL fault simulation using decision diagrams and bitwise set operations," *Proceedings of the 2011 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems*, Vancouver, BC, 2011, pp. 164-170, https://doi.org/10.1109/DFT.2011.42.

[26] R. Ubar, S. Devadze, J. Raik and A. Jutman, "Fast fault simulation for extended class of faults in scan path circuits," *Proceedings of the 2010 Fifth IEEE International Symposium on Electronic Design, Test & Applications*, Ho Chi Minh City, 2010, pp. 14-19, https://doi.org/10.1109/DELTA.2010.32.

[27] U. Reinsalu, J. Raik and R. Ubar, "Register-transfer level deductive fault simulation using decision diagrams," *Proceedings of the 2010 12th Biennial Baltic Electronics Conference*, 2010, pp. 193-196, https://doi.org/10.1109/BEC.2010.5631842.

[28] D. B. Armstrong, "A deductive method for simulating faults in logic circuits," *IEEE Transactions on Computers*, vol. C-21, no. 5, pp. 464-471, 1972, https://doi.org/10.1109/T-C.1972.223542.

[29] N. Vinod et al., "Performance evaluation of LUTs in FPGA in different circuit topologies," *Proceedings of the 2020 International Conference on Communication and Signal Processing (ICCSP)*, 2020, pp. 1511-1515, https://doi.org/10.1109/ICCSP48568.2020.9182074.

***ANNA HAHANOVA** was born in 1978 in Ukraine. Ph.D., Associate Professor of Design Automation Department, Computer Engineering Faculty, Kharkov National University of Radioelectronics, Ukraine. R&D fields: Cyber-physical, cyber-social computing, pattern recognition and machine learning. Digital Smart Cyber University. Previous positions: Deputy Dean of Computer Engineering Faculty (2013-2016). Author of more than 85 publications and 4 monographs, 1 patent and 39 articles indexed in Scopus; 93 citations by 83 documents, h-index = 7.*