# Features of Decision Support's Program at Choice of Tests Optimized Sequence for Semiconductors Memory Diagnosing

Ryabtsev V.G, Almadi M.K.

*Abstract* **– a method, which allows decreasing calculations works labour intensiveness at the choice of paret-optimum tests for semiconductors storage devices diagnosing is offered. This method is realized in the program Optimal-test, which allows decreasing the duration of memory's microcircuits testing without worsening of their quality.**

*Index Terms* **– diagnosing, microcircuit, memory, test**.

## I. INTRODUCTION

At mastering mass-produced storages devices it is necessary to choose a control-diagnostic equipment and set of effective tests to include them in the program of tests. Acquisition of control-diagnostic facilities is carried out on the basis of analysis of technical descriptions and determination of their conforming to produced requirements. First task behave to the cleanly technical tasks, its decision does not require considerable efforts. Considerable difficulties arise up at the choice of effective diagnostic tests and modes task of tests of semiconductor memory's wares [1-3].

For providing of the production plan fulfilling and issue of high quality wares, it is necessary to apply the optimized sequence of tests, which provide achieving high efficiency of diagnosing at the limited resources of productions of time and equipment. For providing of operative adjustment of the program of tests a programmatic facilities development of decision-making support is required.

To improve the wares quality it is necessary to multiply the duration of the test diagnosing, that will multiply the cost of wares and diminishes the Sales Quantity, and the circle of causal-effective connections is locked. Every firm decides independently, what time of the test diagnosing will be for it optimum and will allow to attain the set quality of wares.

Providing of possible economic level of memory's microcircuit and modules diagnosing of memory is one of tasks, which must be decided by managers of technical quality control departments of firms of designing, making and exploiting facilities of the computing engineering, which are intended for the informative and control systems construction.

For diminishing of labour intensiveness of solving of this task, a special methods, algorithms and programs development is needed, which allow to use wide possibilities of the modern computing engineering for sorting of data array, describing a priori properties of the applied tests.

In the known algorithms of data sorting the arrays of integer or real numbers are processed and a new well-organized by ascending array of numbers is formed as a result [4]. However these algorithms after implementation of sorting operation do not allow identifying property, which objects estimate well-organized cells of got array.

The purpose of this paper is development of algorithm and method of choice of the optimized set of tests, providing combination of high efficiency with the possible economic level of memory's microcircuit and modules diagnosing of memory.

## II. MATHEMATICAL MODEL AND SOLVING OF RESEARCH TASK

At the choice of the tests optimized sequence for memory's microcircuits and modules diagnosing, it is necessary to take into account, that their diagnostic properties are unclear certain, and tests are offered, which their detailed algorithmic description is not expounded even. In the conditions of presence of a priori unclear information about diagnostic properties of tests it is expedient to apply the method of choice of paret-optimum tests, in which criterion of tests quality determined by skilled specialists in area of semiconductors storages devices diagnosing.

The most essential criterion for the estimation of tests properties is their ability to find out most widely showing up refusals of memory's cells, decoders of address, reading amplifiers of charges renewal charts in memorizing condensers and other.

These properties of tests are estimated by probabilities of finding out the refusals of the set types and measured by real numbers in a range from 0 to 1 and therefore they compared easily.

The vector of comparison $S_i^{\vartheta\mu}$ of tests properties $\vartheta$ and $\mu$ on probably of finding out the refusal of $i$-type is determined by the following expression:

Ryabtsev V.G. is with the Cherkassy state technological university, Boulevard Shevchenko, 460, CHSTU, Cherkassy, Ukraine. 18006. Tel. (+380)-472-730271. E-mail: volodja18@ukr.net

Al Madi Mudar . is with the Cherkassy state technological university, Boulevard Shevchenko, 460, CHSTU, Cherkassy, Ukraine. 18006. Tel. (+380)-472-730271.E-mail: mudarinfo@yahoo.com.

$$S_i^{\vartheta\mu} = 1, \text{if } q_{i_i}^{\vartheta} > q_{i_i}^{\mu};$$

$$S_i^{\vartheta\mu} = 0, \text{if } q_{i_i}^{\vartheta} = q_{i_i}^{\mu};$$

$$S_i^{\vartheta\mu} = -1, \text{if } q_{i_i}^{\vartheta} < q_{i_i}^{\mu},$$

where $q_{i_i}^{\mu}, q_{i_i}^{\vartheta}$ – probabilities of finding out the refusals of $i$ type of tests $\vartheta$ and $\mu$ accordingly.

For every criterion it is possible to get its mean value by the formula: $E_{mid}^j = \sum\limits_{i=1}^{n} E_{ij} / n$

Then the rationed value of estimation $C_{ij}$ of properties of i- test on finding out the malfunction of j-type it is possible to get through this expression:

$$C_{ij} = 1, \text{if } E_{ij} > E_{mid}^j;$$

$$C_{ij} = 0, \text{if } E_{ij} = E_{mid}^j;$$

$$C_{ij} = -1, \text{if } E_{ij} < E_{mid}^j.$$

An important factor is also time of tests implementation, which can depending on complication of test be measured in microseconds, hours and even to achieve astronomical values. Logically, that tests of which duration do not accord the economic feasibilities of production or exploitation conditions $t_j > t_{max}$, it is necessary to exclude from the examined great number.

For the remaining group, it is necessary to ration the value of parameter testing time and for every test to get the vector of estimation of properties on execution time in according to expression: $S_t^j = (\sum\limits_{j=1}^{n} t_j / n)/t_j$, where

$S_t^j$ – estimation of properties of $j$-type test at implementations times; $n$ – Number of tests; $t_j$ – time of implementation of $j$-test.

Estimations for $m$ criteria and $n$ tests are presented in table. 1.

TABLE 1
ESTIMATIONS FOR TESTS PROPERTIES

| Test Name | Estimations for criteria's | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | … | m |
| Test 1 | $S_{11}$ | $S_{12}$ | $S_{13}$ | … | $S_{1m}$ |
| Test 2 | $S_{12}$ | $S_{22}$ | $S_{23}$ | … | $S_{2m}$ |
| | | … | | | |
| Test $n$ | $S_{n1}$ | $S_{n2}$ | $S_{n3}$ | … | $S_{nm}$ |

Generalized estimation $S_{sum}^i$ for every test it is possible to get by the formula: $S_{sum}^i = \sum\limits_{j=1}^{m} s_{ij}$.

Thus, the task of choice of the optimized tests set is taken to ranging of tests, or to the sorting by their generalized estimations ascending.

## III. FEATURES OF THE PROGRAM OF THE OPTIMIZED TESTS SEQUENCE CHOICE

For the programming realization of researched task it is suggested to use the determined by the user structured information with the following kind:

```
Typedef struct
{
  Int Estimation;
  Int Number;
  Char Name [12];
} DATA;
```

As input data to the computer's memory the array of the structured data, containing probabilities of exposure of refusals and tests names, is added.

For sorting of data array it is suggested to use the algorithm of "bubble", where records with the"easy" values of the key field emerge upwards like a bubble. With the purpose of diminishing of steps of algorithm a special flag, allowed comparison of array cells, is entered, and in the beginning this value of this flag will be one.

After activation of comparison operation of a flag with a value zero, and then in pairs compare array cells: first with the second, second with the third and etc, simultaneously assort them. If transposition of elements was carried out, a value is again the flag appropriates a value one and a new cycle of elements sorting begins. If a flag was not set in the one state, it means that there is nothing to assort and a process is stopped.

## IV. PRACTICAL RESULTS OF RESEARCH

Diagnostic experiments were executed at two different values of power supply tension: high tension (HVcc) and low tension (LVcc). At HVcc in 1545 microcircuits were discovered disrepairs, from them 1343 microcircuits the defects were detected by all tests and only in 202 microcircuits the defects were discovered only by one or a few tests [5].

For implementation of diagnostic experiments the tests were used, of which execution time for the memory's microcircuits of different capacity is resulted in tabl1.

Probabilities of finding out refusals in the microcircuits of memory through the sets of tests of consisting of combinations from two tests calculated on formulas:

$$\forall i, i = \overline{1,n}, q_i = \frac{m_i}{m_s}, \forall i, j = \overline{1,n}, i \neq j, q_{ij} = \frac{m_i \cup m_j}{m_s},$$

where $m_i$, $m_j$ – an amount of microcircuits, discovered by the tests of $i$, $j$ accordingly; $m_s$ – common amount of defective microcircuits.

Implementation time of the most widespread tests is resulted in table. 2. Time of implementation of different tests can be measured by seconds and can achieve a few days even.

If not to apply measures on optimization of tests set, so the time of arbitrary sequence of tests performance can reach great values and thus the cost of memory's microcircuits will grow sharply.

TABLE 2

IMPLEMENTATION TIME OF EXAMINED TESTS

| № | Test Name | Formula | Time while tc = 5 nsec and memory's capacity Mb, sec | | | | |
|---|-----------|---------|------|------|------|------|------|
| | | | 1 | 4 | 16 | 64 | 256 |
| 1 | WalkRow | $8n+2nC$ | 10.779 | 86.067 | 687.866 | 5500 | 43990 |
| 2 | WalkColumn | $8n+2nR$ | 10.779 | 86.067 | 687.866 | 5500 | 43990 |
| 3 | GalRow | $6n+4nC$ | 21.508 | 171.925 | 1375 | 11000 | 87970 |
| 4 | GalColumn | $6n+4nR$ | 21.506 | 171.925 | 1375 | 11000 | 87970 |
| 5 | Hammer | $49n$ | 0.257 | 1.028 | 4.11 | 16.442 | 65.767 |
| 6 | March SL | $41n$ | 0.215 | 0.86 | 3.439 | 13.757 | 55.029 |
| 7 | March RAW | $26n$ | 0.136 | 0.545 | 2.181 | 8.724 | 34.897 |
| 8 | March SS | $22n$ | 0.115 | 0.461 | 1.845 | 7.382 | 29.528 |
| 9 | March G | $23n$ | 0.121 | 0.482 | 1.929 | 7.718 | 30.87 |
| 10 | March SR | $14n$ | 0.073 | 0.294 | 1.174 | 4.698 | 18.79 |
| 11 | PMOVI | $13n$ | 0.068 | 0.273 | 1.091 | 4.362 | 17.448 |
| 12 | March C | $10n$ | 0.052 | 0.21 | 0.839 | 3.355 | 13.422 |
| 13 | MATS++ | $6n$ | 0.031 | 0.126 | 0.503 | 2.013 | 8.053 |
| 14 | MATS+ | $5n$ | 0.026 | 0.105 | 0.419 | 1.678 | 6.711 |
| 15 | Scan | $4n$ | 0.021 | 0.084 | 0.336 | 1.342 | 5.369 |

Results of choice of Paret-optimum sequence of tests, got on the offered algorithm, at HVcc resulted on a fig. 1.


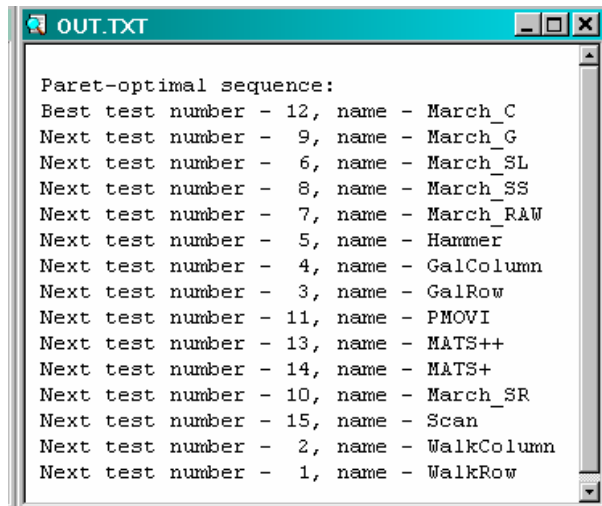
Fig. 1. Paret-optimum sequence of tests at HVcc

By results of experimental researches it is established, that 6 tests from the optimized set find out all defects of the given microcircuits, therefore the application of other tests is inexpedient.

The graphs of change duration of memory's microcircuits diagnosing are with a capacity 4 Mb at time of access cycle, equal 5 nsec at HVcc for the arbitrary sequence of tests and optimized sequence resulted on a fig. 2.

Implementation of 6 tests time from an arbitrary set of tests is 345,675 sec and for 6 tests from the optimized sequence – only 3,586 sec.
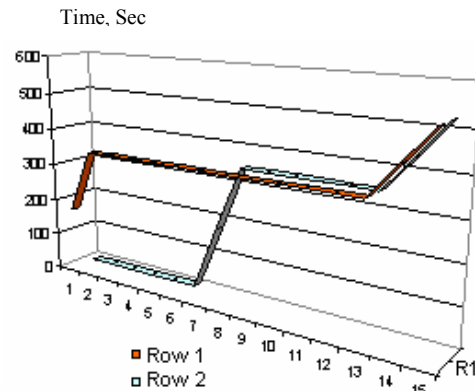


Fig. 2. Duration of diagnosing for the different sets of tests:
Row 1 - for the arbitrary sequence of tests;
Row 2 - for the optimized sequence

The histogram of duration of diagnosing for 6 tests from different sets is resulted on a fig. 3.
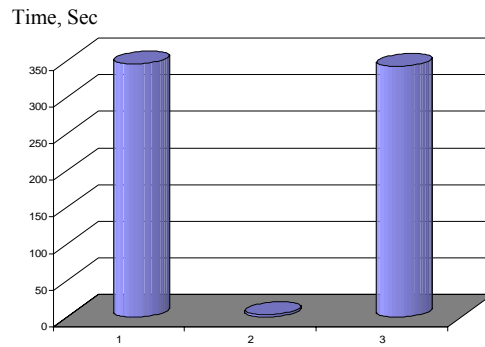


Fig. 3. Duration of diagnosing for different sets from 6 tests:
Column 1 – for the arbitrary sequence of tests; Column 2 – for the optimized sequence; Column 3 – economy of time.

As a result of analysis of this histogram, it is possible to do a conclusion, that the economy of time at application of the optimized set of tests makes 342,089 sec.

The results of choice of Paret-optimum sequence of tests at LVcc are resulted on a fig. 4.



```
OUT.TXT                                    _ □ ×

Paret-optimal sequence:
Best test number - 12, name - March_C
Next test number -  5, name - Hammer
Next test number -  3, name - GalRow
Next test number -  7, name - March_RAW
Next test number -  4, name - GalColumn
Next test number -  9, name - March_G
Next test number - 11, name - PMOVI
Next test number -  6, name - March_SL
Next test number -  8, name - March_SS
Next test number - 14, name - MATS+
Next test number - 13, name - MATS++
Next test number - 15, name - Scan
Next test number - 10, name - March_SR
Next test number -  2, name - WalkColumn
Next test number -  1, name - WalkRow
```
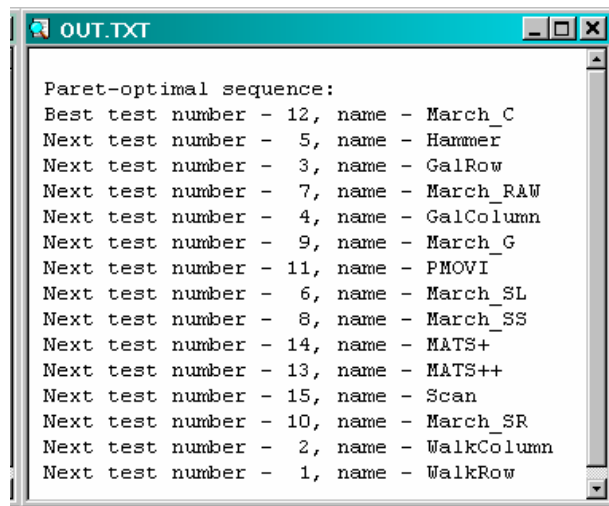
Fig. 4. Paret-optimum sequence of tests at LVcc

The graphs of change duration of memory's microcircuits diagnosing are with a capacity 4 Mb at time of access cycle, equal 5 nsec at LVcc for the arbitrary sequence of tests and optimized sequence resulted on a fig. 5.
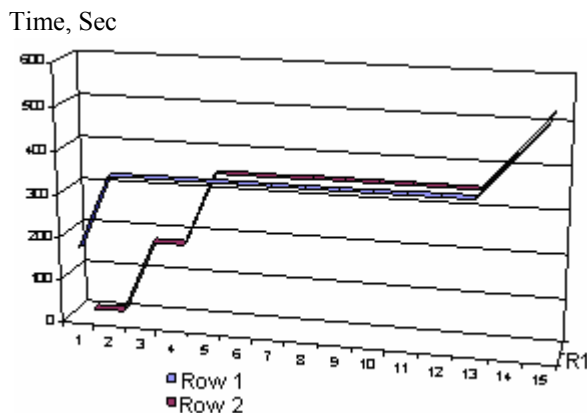
Time, Sec



Fig. 5. Duration of diagnosing for the different sets of tests:
Row 1 – for the arbitrary sequence of tests;
Row 2 – for the optimized sequence

The obtained results show that for the different values of tensions of power supplies it is necessary to apply the different sets of tests, providing most economic expenses on conducting of the test diagnosing.

For two combinations of power supply tension: HVcc and LVcc time of diagnosing through 6 tests for the arbitrary sequence of tests is 691, 35 sec (see fig. 6), and for the optimized sequence of tests only 349,701 sec. Thus, the economy of time due of use of Optimal-test program is about 50%.
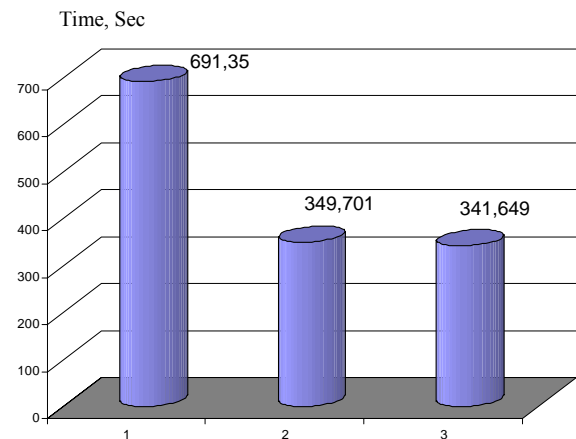
Time, Sec



Fig. 6. Histogram of duration of tests:
First column – execution time of 6 tests from an un optimized set,
Second – execution time of 6 tests from an optimized set,
Third – economy of time

## V. CONCLUSION

The offered method and program allow decreasing labour intensiveness of calculations works at the choice of Paret-optimum tests for diagnosing of semiconductors storages devices. Application of the program Optimal-test will allow reducing expenses on implementation of the test diagnosing of storage devices and decrease their prime price.

REFERENCES

[1] *Van de Goor, A.J.*: 'Testing semiconductor memories, theory and practice' (ComTex Publishing, Gouda, The Netherlands, 1998, 2nd edn.)
[2] *Melnikov A.V., Ryabtsev V.G.* Control of computers memory's modules. – K.: "Kornichuk", 2001. – 172 p.
[3] *Ryabtsev V.G., Kudlaenko V.M, Movchan Y.U.* Method of estimation diagnostic properties of the Test Family March. // Proceeding of East-West and Test International Workshop (EWDTW'04). Yalta, Alushta, Crimea, Ukraine, September 23-26, 2004. – Pp. 220-224.
[4] *Aho A.V., Hopkroft D.E., Ulman D.D.* Data structure and algorithms. – M.: Publishing House "Williams", 2000. – 384 p.
[5] *Hamdioui S., Van de Goor Ad J., Reyes D. J., Rodgers M.* Memory test experiment: industrial results and data. // IEE Proc.-Computer. Digit. Tech., Vol. 153, No. 1, January 2006. – Pp. 1-8.

**Ryabtsev Volodymyr,** doctor of engineering sciences, professor of information technologies and systems faculty, in Cherkassy state technological university. Scientific interests: diagnosing of semiconductor storages devices, instrumental tools for the synthesis of tests of digital systems models verification, which contain built-in memory. Address: Boulevard Shevchenko, 460, CHSTU, Cherkassy, Ukraine. 18006. Tel. (+380)-472-730271. E-mail: volodja18@ukr.net

**Al Madi Mudar**, postgraduate student of the Cherkassy state technological university. Scientific interests: Verification of memory's microcircuits models. Address: Boulevard Shevchenko, 460, CHSTU, Cherkassy, Ukraine. 18006. Tel. (+380)-472-730271. E-mail: mudarinfo@yahoo.com.

# General Testing Models of SOC Hardware-Software Components

Vladimir Hahanov, *IEEE Computer Society Golden Core Member*, Eugenia Litvinova, *IEEE Society member, W. Gharibi*

*Abstract* — **Innovative testable design technologies of hardware and software, which oriented on making graph models of SoC components for effective test development and SoC component verification, are considered. A novel approach to evaluation of hardware and software testability, represented in the form of register transfer graph, is proposed. Instances of making of software graph models for their subsequent testing and diagnosis are shown.**

*Index Terms* — **Infrastructure Intellectual Property, Register Transfer Graph, System-on-a-Chip, Testing.**

## I. HARDWARE-SOFTWARE TESTABILITY

ADAPTATION of testing and verification methods of digital systems can bring in big financial and time dividends, when using for testable design and diagnosis of software. Consideration of the following questions can be interesting: 1. Classification of key uses of SoC testable design technologies in software testing and verification problems. 2. Universal model of hardware and software component in the form of directed register transfer and control graph, on which the testable design, test synthesis and analysis problems can be solved. 3. Metrics of testability (controllability and observability) evaluation for hardware and software by the graph register transfer and control model.

The silicon chip that is basis of computers and communicators development has to be considered as the initiate kernel of new testing and verification technologies appearance in software and computer engineering. A chip is used as test area for new facilities and methods creation and testing for component routing, placement, synthesis and analysis. Technological solutions, tested by time in microelectronics, then are captured and implemented into macroelectronics (computer systems and networks). Here

are some of artifacts, relating the continuity of technological innovations development:

1. The Boundary Scan Standards [1] for board and chip levels result in the assertion technique appearance for software testing and verification. 2. The testability analysis facilities [2] (controllability and observability) of digital structures can be adapted to the evaluation of software code to detect critical statements and then to improve software relative to the testability criteria. 3. The covering analysis technologies [3] for given faults by test patterns have to be used for making of the fault covering table of software to estimate the test validity and to diagnose. 4. The Thatte-Abraham [4] and Sharshunov [5] graph register transfer models have to be used for software testing that is reduced to more technological form by structural-logical analysis. 5. Partition of an automaton on control [2] and operating parts is used for reduction of software verification on basis of preliminary synthesis of control and data transfer graphs. 6. Lifecycle curve for hardware [6] represents time stages of yield change at creation, replication and maintenance of software. 7. Platform-based electronic system-level design [7] by using of existent chip sets and GUI-based is isomorphic to the object-oriented programming technology on basis of created libraries. Application of the Electronic System Level Technology in the programming enables to use finished software functional components from basic libraries to create new software. In this case the main design procedure is mapping, oriented on covering of specification functions by existent components, at that new code is nothing more than 10% of a project. 8. The testbench notion [8] that is used for hardware testing and verification by means of HDL-compilers appears in software, realized on C++ language level and higher. 9. Platform-based testbench synthesis [7] by using the existent test libraries (ALINT) for components – standardized GUI-based F-IP SoC functionalities. It has to be used for software test generation on basis of developed libraries of the leading companies. 10. Standard solutions of F-IP in the framework of I-IP [9] can be used for embedded software component testing including faulty software module repair. 11. Two-dimensionality assurance in a structure of interconnected functional components (IP-cores) of developed software is based on use of multicore architectures for technological paralleling of computational processes [10] that is quite urgent in the conditions of technological revolution, proposed Intel. 12. Creation of address space for SoC functionalities, which are realized as hardware or software,

gives a digital system the marvelous self-repair feature by means of I-IP for hardware and software components. An instance of it is robust multicore version of hardware. At that a faulty addressable component can be replaced by other one (faultless) in the process of operation. Addressability has to be used when creation of critical software, in which availability of addressable diversion (multiversion) components gives a software system an opportunity to replace components at fault appearance. 13. The technological problem of offline on-chip self-testing, self-diagnosis and self-repair by using external facilities (or without them), which are solved by all leading companies, is quite interesting. To solve the problem the modern wireless and Internet technologies of distant service are applied. Disadvantage of these technologies is opportunity of remote unauthorized access to a chip that can result in unwanted destructive consequences and digital system failure. Though, the specific character of digital system-on-a-chip is the marvelous ability to remove faults distantly due to chip connection with outer space by means of Internet or wi-fi, wi-max, bluetooth technologies, which are realized on a chip. Distance correction of software errors is possible due to utilization of SoC memory (which occupies up to 94% of chip area) for software storage. In a case of error detection new faultless code can be saved to this memory. Distance correction of hardware errors is possible due to utilization of Erasable Programmable Logic Device (EPLD), where new faultless bit stream can be saved in a case of fault detection; actually thereby new hardware is created by means of chip reprogramming.

Approximation and interpenetration of technologies result in isomorphic design, testing and verification methods in relation to software and hardware complexes that in essence are natural process of progressive concept assimilation. The most important characteristics of product lifecycle (time-to-market and yield) become commensurable by time and production volume and this fact favours the tendency above. The hardware lifecycle curve, shown in Fig. 1, to within the isomorphism represents time software stages: design, production ramp-up, fabrication improvement and maintenance.
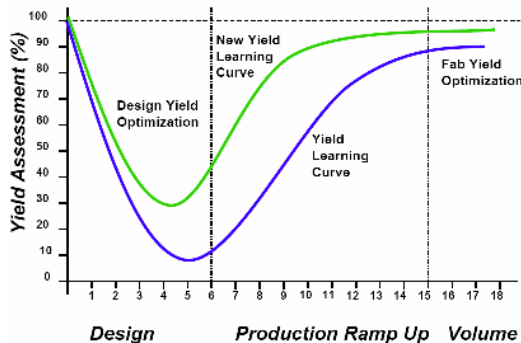

Fig. 1. Lifecycle curve of hardware-software complex

In the context of lifecycle there are two urgent problems relating to a curve lifting ordinate-direction and a curve compression time-direction that means time-to-market reduction. Here yield rise takes place on all stages: design – because of design errors recovery, production ramp up –

correction of code, implemented to SoC memory, volume – because of service pack release, which correct errors by means of distribution by Internet or satellites.

The research aim is to show development directions of effective testable design models and methods for software to raise yield by adaptation of hardware design technologies and reduction of software structures to the existent standards and patterns of testing and verification. The research problems: 1) Development of a software model for testable design and verification; 2) Development of software testing and diagnosis technologies on basis of the register models of operational and control software parts.

## II. SoC SOFTWARE TESTING TECHNOLOGIES

The standard IEEE 1500 SECT [1] has to be considered as effective component of SoC Infrastructure Intellectual Property. The main its destination is testing of all F-IP functionalities and galvanic connections between them. Next step in evolution of the standard for the purpose of repairable chip creation is development of I-IP components with SoC diagnosis and repair service functions; last ones in the aggregate with a testing module are market attractive: $I = \{T, D, R\}$. The diagnosis and repair procedures are not regulated by the testable design standards because of the complexity and ambiguity of a universal solution of this problem for various types of computers. For irregular or unique structures solutions of all three problems are based on a priori redundancy – diversification of component functionalities, which make up SoC. At that rate only it can to say about on-chip repair of a fail element. Concerning regular structures, which have underlying redundancy, such as multi- and matrix processors, one of solution variants can be a controller structure that combines realization of all functions above by means of the Boundary Scan Standard:

$$I = \{T, F, S, D, R\}, \ T = \{T^1, T^2, ..., T^i, ..., T^n\};$$
$$F = \{F^1, F^2, ..., F^i, ..., F^n\};$$
$$S = \{S^1, S^2, ..., S^i, ..., S^n\}; \ D = f(T, F, S) = F^D \in F;$$
$$R = g(D, F) = (F^R \subseteq F) \& (F^R \cup F^D = F).$$

Here the first three identifiers of a model are tests for functionalities; components, which represent functions; and boundary scan register cells for identification of functionalities' technical state. Other two ones are represented by functions for SoC diagnosis and repair realization. The first function (D) defines a faulty components set that is computed on basis of the output response vector S and a test, covered all functional faults; it is entered in the form of fault detection table (FDT). Second function (R) formulates the rules of component power reduction by removal of fault elements from addressing and forming of new faultless subset F-IP SoC to use according to its intended purpose.

A question about location of a test and functionality verification analyzer is not problematical. If the matter is unique components they should be connected with service I-IP components no dispersal on a chip area. In a case of the

regular matrix structure tests for all cells are the same, so it has to be used for all components; also a single test, diagnosis and repair analyzer has to be in a structure.

A question about computer resource relocation after a faulty cell detection is interesting. If there are additional spares for repair, the problem comes to the optimal replacement of faulty memory cells by spare rows and columns. In other case there are other system repair models, which depends on a multiprocessor representation form. The linear or one-dimensional addressing form defines consistency of input variables n of a decoder and addressable components, which are connected among themselves by relation: $|A| = 2^n$. The matrix representation of a multiprocessor specifies two-dimensional component addressing that is oriented on pipelining technologically. In both cases decoding of a cell number by its address is carried out. So, for the purpose of a faulty component address change on a faultless one it is necessary to modify a decoder structure. This problem is strictly technical and its solution comes to the masking of faulty component addresses. Other solution is related to availability of spares in a processor structure. In given case the problem can be reduced to the replacement of one or several faulty processors by faultless elements from the spare. The optimal solution of the problem has considered for a case, when there are several faulty cells in a memory matrix. The problem becomes more complex if digital system functionality has parallelized yet on existent processor matrix $P = P_{ij}, |P| = m \times n$, which has faulty elements, and it is necessary to reallocate a set of faultless cells $|P*| \leq |P|$ to obtain the quasioptimal covering of functional subproblems by a subset of faultless processors.

Development of software formal model, to which CAD and EDA technologies can be applied, to use the formal methods of test synthesis, evaluation of fault covering, determination of testability (controllability and observability) for subsequent modernization of software structure is quite urgent. To solve this problem the automaton model cab be used:

$$\begin{cases} M = (M^{OA}, M^{CA}), M^{OA} = \{\vec{X}^O, Y^C, Y^O, \vec{Z}^O\}; \\ M^{CA} = \{X^C, Y^O, Y^C, Z^C\}; \\ \vec{Z}^O = f^O(\vec{X}^O, Y^C); Y^O = g^O(\vec{X}^O, Y^C); Z \\ C = f^C(X^C, Y^O); Y^C = g^C(X^C, Y^O). \end{cases}$$

Where $\vec{X}^O, \vec{Z}^O$ are vectors or register input and output variables; $Y^C, Y^O, Z^C$ are signals of operation control (initialization), announcing signals, and monitoring signals of a control automaton respectively; $f^O, g^O (f^C, g^C)$ are functions, which determine relations between interface signals in an operational and control automata.

But the automaton model above $M = (M^{OA}, M^{CA})$ is not technological for a developer at solution of practical

problems of testable design. Processor (software)-based modification of one is proposed; it consists of two graphs with directed ribs:

$$\begin{cases} M = (M^{OR}, M^{CG}), M^{OR} = \{R, I\}; \\ R = \{R_1, R_2, ..., R_i, ..., R_n\}, I = \{I_1, I_2, ..., I_j, ..., I_m\}; \\ R_i = f(R_k, I_j); M^{CG} = \{S, E\}; \\ S = \{S_1, S_2, ..., S_i, ..., S_p\}, E = \{E_1, E_2, ..., E_j, ..., E_q\}; \\ S_i = f(S_k, E_j). \end{cases}$$

Here $M^{OR}$ is Sharshunov register transfer graph [5] with a set of points R, which describes all memory components (registers, flip-flops, counters, memory, input and output buses) used in a program, and a set of ribs, which are marked by instructions I and activate information transfer between points. Expression $R_i = f(R_k, I_j)$ defines functional dependence between adjacent points $R_i \rightarrow R_k$, which are connected by means of operation $I_j \in I$.

Component $M^{CG}$ is conceptual graph of a control automaton that is defined on a point set S, which are connected by directed ribs E, marked by transition conditions. Expression $S_i = f(S_k, E_j)$ defines functional dependence between adjacent points $S_i \rightarrow S_k$ of a control graph, which are connected to realize jump $E_j \in E$.

Instances of register transfer and control graphs are shown in Fig. 2 and 3 respectively.
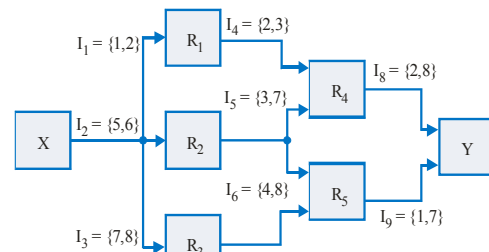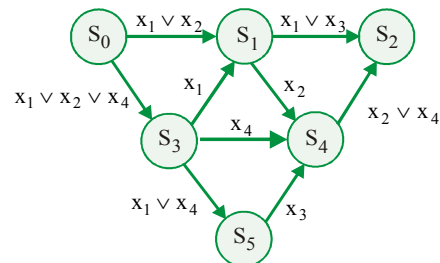


Fig. 2. Register transfer graph



Fig. 3. Control automaton graph

Advantages of graph models are not only in structure representation of functionals interaction, but applicability of testability analysis methods, because directed graph models have explicit information flow directions, input and output points. On the basis of the testability evaluation experience for digital systems the following metrics of controllability

and observability analysis for the graph structures above can be proposed:

$$G = \{R, I\}; \quad R = \{R_1, R_2, ..., R_n\}, I = \{I_1, I_2, ..., I_m\};$$

$$I_{ij} \in I_i \approx (R_p R_q);$$

$$C(R_q) = \frac{1}{k} \times \sum_{i=1}^{k} \left[ \frac{1}{m} \times \left| \bigcup_j I_{ij} \in (R_p R_q) \right| \times C(R_p) \right];$$

$$O(R_p) = \frac{1}{k} \times \sum_{i=1}^{k} \left[ \frac{1}{m} \times \left| \bigcup_j I_{ij} \in (R_p R_q) \right| \times O(R_q) \right];$$

$$C(R_x) = 1; \quad O(R_y) = 1.$$

Here a software (hardware) module model is represented by the graph $G = \{R, I\}$ that consists of points (registers) and ribs (instructions). Every graph rib is marked not less one operation $I_{ij} \in I_i \approx (R_p R_q)$ that forms a command subset, attached to the rib $(R_p R_q)$. The controllability criterion for the point $C(R_q)$ depends on the controllability of previous point $C(R_p)$ and reduced additive power of a command set

$$\frac{1}{k} \times \sum_{i=1}^{k} \left[ \frac{1}{m} \times \left| \bigcup_j I_{ij} \in (R_p R_q) \right| \right] = \frac{1}{k} \times \sum_{i=1}^{k} \left[ \frac{1}{m} \times d \right] = \frac{1}{k} \times \sum_{i=1}^{k} \left[ \frac{d}{m} \right],$$

which activate k ribs, attached to the given point $C(R_q)$. Here every rib contains d operations (m – the total command quantity), which initiate information transfer to $(R_p R_q)$. On the analogy the observability evaluation criterion $C(R_p)$ based on analysis of points-successors and ribs, outgoing from $C(R_p)$, is formed. The advantage of the proposed models and criteria of controllability and observability evaluation is their universality, based on realization of direct and inverse implication on a graph, as well as their invariance concerning the testable analysis and test synthesis for software and hardware components. Controllability $C(R_x) = 1$ of input and observability $O(R_y) = 1$ of output graph points is initiated by "1" values. As advancement of point analysis to internal lines the values of evaluations above can decrease only.

Thus, the graph points are represented by the following components: input variables, output variables, register variables, ALU block, memory arrays, which are represented in a format of their presentation in a software (hardware) module. Ribs determine an operand (command) set, which transfer (transformation) of information between points. The complete model of a device, represented by the register transfer and control graphs, covers all statements of data transfer and control in a software (hardware) module that is necessary to the synthesis of testable device. At that test synthesis is based on solving of the covering problem of all paths and points in register transfer and control graphs by testbench statements.

The integral evaluation of the point testability in a graph is calculated by formula: $T(R_i) = C(R_i) \times O(R_i)$.

The total graph testability for software (hardware) is computed by expression $T_{total} = \frac{1}{n} \sum_{i=1}^{n} T(R_i)$.

For instance, represented by a register transfer graph (Fig. 2), computation of testability is given below. The controllability factors are:

$$C(X) = 1;$$

$$C(R_1) = C(X) \times 1 \times \frac{d}{m} = 1 \times 1 \times \frac{2}{8} = \frac{2}{8} = 0,25;$$

$$C(R_2) = C(X) \times 1 \times \frac{d}{m} = 1 \times 1 \times \frac{2}{8} = \frac{2}{8} = 0,25;$$

$$C(R_3) = C(X) \times 1 \times \frac{d}{m} = 1 \times 1 \times \frac{2}{8} = \frac{2}{8} = 0,25;$$

$$C(R_4) = \left( \left[ C(R_1) \times \frac{d}{m} \right] + \left[ C(R_2) \times \frac{d}{m} \right] \right) \times \frac{1}{2} = \frac{1}{16} = 0,0625;$$

$$C(R_5) = \left( \left[ C(R_2) \times \frac{d}{m} \right] + \left[ C(R_3) \times \frac{d}{m} \right] \right) \times \frac{1}{2} = \frac{1}{16} = 0,0625;$$

$$C(Y) = \left( \left[ C(R_4) \times \frac{d}{m} \right] + \left[ C(R_5) \times \frac{d}{m} \right] \right) \times \frac{1}{2} = \frac{1}{64} = 0,015625;$$

The point Y has minimal controllability. Observability computation:

$$O(Y) = 1; \quad O(R_4) = O(Y) \times \frac{2}{8} = 1 \times \frac{2}{8} = 0,25;$$

$$O(R_5) = O(Y) \times \frac{2}{8} = 1 \times \frac{2}{8} = 0,25;$$

$$O(R_3) = O(R_5) \times \frac{2}{8} = \frac{1}{4} \times \frac{2}{8} = 0,0625;$$

$$O(R_1) = O(R_4) \times \frac{2}{8} = \frac{1}{4} \times \frac{2}{8} = 0,0625;$$

$$O(R_2) = \left[ \left( O(R_4) \times \frac{2}{8} \right) + \left( O(R_5) \times \frac{2}{8} \right) \right] / 2 = \frac{1}{16} = 0,0625;$$

$$O(X) = \left[ \left( O(R_1) \times \frac{2}{8} \right) + \left( O(R_2) \times \frac{2}{8} \right) + \left( O(R_3) \times \frac{2}{8} \right) \right] / 3 =$$

$$= \frac{1}{64} = 0,015625.$$

The point X has minimal observability. Testability computation:

$$T(X) = 1 \times 0,015625 = 0,015625;$$
$$T(R_1) = 0,25 \times 0,0625 = 0,015625;$$
$$T(R_2) = 0,25 \times 0,0625 = 0,015625;$$
$$T(R_3) = 0,25 \times 0,0625 = 0,015625;$$
$$T(R_4) = 0,0625 \times 0,25 = 0,015625;$$
$$T(R_5) = 0,0625 \times 0,25 = 0,015625;$$
$$T(Y) = 0,015625 \times 1 = 0,015625.$$

The total circuit testability $T_{total} = 0,015625$. Calculation of the testability characteristics for the control automaton graph (Fig. 3) is realized similarly. Determination of the graph controllability:

$$C(S_0) = 1;$$

$$C(S_1) = \left(\left[C(S_0) \times \frac{d}{m}\right] + \left[C(S_3) \times \frac{d}{m}\right]\right) \times \frac{1}{2} = \frac{11}{32} = 0,34375;$$

$$C(S_3) = C(S_0) \times 1 \times \frac{d}{m} = 1 \times 1 \times \frac{3}{4} = \frac{3}{4} = 0,75;$$

$$C(S_5) = C(S_3) \times 1 \times \frac{d}{m} = \frac{3}{4} \times 1 \times \frac{2}{4} = \frac{6}{16} = 0,375;$$

$$C(S_4) = \left(\left[C(S_1) \times \frac{d}{m}\right] + \left[C(S_3) \times \frac{d}{m}\right] + \left[C(S_5) \times \frac{d}{m}\right]\right) \times \frac{1}{3} =$$

$$= \frac{47}{384} = 0,1224;$$

$$C(S_2) = \left(\left[C(S_1) \times \frac{d}{m}\right] + \left[C(S_4) \times \frac{d}{m}\right]\right) \times \frac{1}{2} =$$

$$= \frac{179}{1536} = 0,11654.$$

The point $S_2$ has minimal controllability. Observability computation:

$$O(S_2) = 1; \quad O(S_4) = O(S_2) \times 1 \times \frac{2}{4} = 1 \times 1 \times \frac{2}{4} = 0,5;$$

$$O(S_5) = O(S_4) \times 1 \times \frac{1}{4} = \frac{1}{2} \times 1 \times \frac{1}{4} = \frac{1}{8} = 0,125;$$

$$O(S_3) = \left[\left(O(S_1) \times \frac{1}{4}\right) + \left(O(S_4) \times \frac{1}{4}\right) + \left(O(S_5) \times \frac{2}{4}\right)\right]/3 =$$

$$= \frac{7}{96} = 0,07292;$$

$$O(S_1) = \left[\left(O(S_2) \times \frac{2}{4}\right) + \left(O(S_4) \times \frac{1}{4}\right)\right]/2 = \frac{1}{8} = 0,125;$$

$$O(S_0) = \left[\left(O(S_1) \times \frac{2}{4}\right) + \left(O(S_3) \times \frac{3}{4}\right)\right]/2 = \frac{15}{256} = 0,05859375.$$

The point $S_0$ has minimal observability. Testability computation:

$$T(S_0) = 1 \times 0,05859375 = 0,05859375;$$
$$T(S_1) = 0,34375 \times 0,125 = 0,04296875;$$
$$T(S_2) = 0,11654 \times 1 = 0,11654;$$
$$T(S_3) = 0,75 \times 0,07292 = 0,05469;$$
$$T(S_4) = 0,1224 \times 0,5 = 0,0612;$$
$$T(S_5) = 0,375 \times 0,125 = 0,046875.$$

The point $S_1$ has the worst testability. The total circuit testability is $T_{total} = 0,063478$.

Thus, the proposed testability evaluation method has the followings advantages: 1) high effectiveness and universality relative to its use for evaluation of register transfer and control graph testability; 2) possibility to detect bottlenecks in software or hardware to modify a project structure; 3) choice of the best project by comparison of alternative variants testability.

### III.  SOFTWARE DIAGNOSIS TECHNOLOGY

At development of large size software verification of development project on the correctness of statements is urgent problem. Complex software includes great many branches and verification of software on every logical path is rather complex problem. A method of faulty statements (errors or faults) searching for software that is based on representation of software algorithm in the form of graph structure for subsequent test generation and fault diagnosis is considered below on an example. Lets it is necessary to verify the software that realizes computation of the following sum of functions:

$$S = (x) + \omega(x),$$

$$x = \begin{cases} x + 3; & x < 2; \\ 2x - 3; & 2 \le x < 12; \\ -3x + 7; & x \ge 12; \end{cases}$$

$$\omega(x) = \begin{cases} \sin(x + \pi/3), x < 2\pi/3; \\ \sin(\pi in + 2, x \ge 2\pi/3. \end{cases}$$

One of the possible problem solution variants on C++ language is represented by the following listing:

*Listing 3.1.*

```cpp
#include <iostream>
#include <math.h>
using namespace std;
int main()
{
    const double Pi=3.14159;
    double F, w, f, x;
    cin>>x;
    if (x<2) f=x+3;
    else if ((x>=2) && (x<12)) f=2*x-3;
    else f=-3*x+7;
    if (x<2./3.*Pi)
    w=sin(x+Pi/3);
    else w=sin(Pi*x)+2;
    F=f+w;
    cout<<F<<endl;
    return 0;
}
```

Lets an error takes place in a statement of computational part of software. Instead of the correct statement

```cpp
else w=sin(Pi*x)+2;
```

the following one is written:

```cpp
else w=sin(Pi*x) 2;
```

It is necessary to detect faulty statement in program code by using the testing technology, based on the graph code model. Software diagnosis stages include 4 procedures below.

1. Making of register transfer graph.

Graph ribs are a set of code fragments or separate operations (Fig. 4); graph points are points of information monitoring (registers, variables, memory), which are used for forming of assertions too.
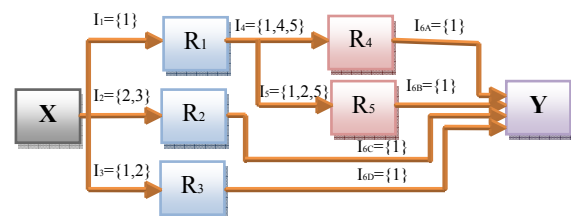


Fig. 4.  Register transfer graph

A number of test points in the graph (registers, variables, memory) should be adequate to diagnose of given resolution. Otherwise it is necessary to carry out the analysis of register transfer graph testability for software and to determine the minimal additional quantity of observation lines for forming of assertions, which enable to detect faulty modules with given diagnosis resolution. Every rib (see Fig. 4) is marked by an arithmetic operation set: {1} – summation; {2} – multiplication; {3} – subtraction; {4} – division; {5} – obtainment of trigonometric sine. In a case when there is a branch in a program a number of outgoing ribs from a point is equal to quantity of adjacent sinks that is formed by branch statements in respective part of a program.

Thus, for the code fragment of the instance:

if (x<2) f=x+3;
else if ((x>=2) && (x<12)) f=2*x-3;
else f=-3*x+7;

there are three ribs, outgoing from the point X. Computational results $I_1, I_2, I_3$, which depend on the variable X, are checked in the points $R_1, R_2, R_3$ respectively. In a case of execution of the operation $I_1$ the following branch is realized:

if (x<2./3.*Pi) w=sin(x+Pi/3);
else w=sin(Pi*x)+2;

Then the general summation operation for all transactions is carried out regardless of which branch statements had been executed.

F=f+w;

The summation operation is executed on various ribs (the objects $I_{6A}, I_{6B}, I_{6C}, I_{6D}$), but all of them correspond to the same part of the program code. So, faultless execution an operation on a rib eliminates a fault on other three ones. On next stages of software diagnosis these objects are merged to $I_6$. The result are checked in the final point Y.

The method of software algorithm representation by graph structure enables to show all possible variants of software execution, as well as to simplify realization of next diagnosis stage of software and forming of minimal test.

2. Test synthesis and analysis. A set of ribs are written in the form of disjunctive normal form (DNF), where every term is one-dimensional path from input port to output, which covers a subset of internal lines: $P = X14Y \vee X15Y \vee X2Y \vee X3Y$. In the aggregate one-dimensional paths, represented in DNF, cover all possible transactions – graph points and ribs. An aggregate of code fragments or statements (activation instructions), written by disjunction, is brought to conformity with every rib. For instance, the path X14Y activates execution of operations on ribs $I_1, I_4, I_{6A}$. At that the ribs $I_1$ and $I_{6A}$ have only one statement, and consecutive execution of three statements corresponds to the identifier $I_4$. The test $P_1 = [(1)(1 \vee 4 \vee 5)(1)]$ that activates the path X14Y ensures the correctness check of all statements. Thus, the test of minimal covering of all graph points and ribs by commands,

which activate graph ribs and therefore data movement to observation points, can be written:

$$P = [(1)(1 \vee 4 \vee 5)(1)] \vee [(1)(1 \vee 2 \vee 5)(1)] \vee$$
$$\vee [(2 \vee 3)(1)] \vee [(1 \vee 2)(1)].$$

Subsequent DNF transformation consists of removal of brackets to obtain complete test that enables to check transactions in a graph, which cover all points and ribs in various combinations:

$$P = (111 \vee 141 \vee 151) \vee (111 \vee 121 \vee 151) \vee$$
$$\vee (21 \vee 31) \vee (11 \vee 12)$$

The obtained test is redundant; it is not always acceptable for large size software, because of there is large quantity of test patterns. So, the ability to create minimal length test of given resolution is very important. Such test is formed by solving of the covering problem of all graph points and ribs and activation of code fragments sets. When testing it is supposed that hardware components, used in the software are faultless.

3. Fault detection table making. Fault detection table is oriented on verification of code fragments sets on ribs, which form data activation paths to the observation points (graph points). In compliance with comparison of experimental data of tested software and expected responses the output response vector V is formed. In a case of result failure on an observed line the respective coordinate of the vector V takes on a value "1" for the test pattern under consideration. The fault detection table of code fragments on complete test $P = X14Y \vee X15Y \vee X2Y \vee X3Y$, where test patterns are written in general form (a set of one-dimensional paths), is shown below:

| $T_i$ / $I_j$ | $I_{11}$ | $I_{22}$ | $I_{23}$ | $I_{31}$ | $I_{32}$ | $I_{41}$ | $I_{44}$ | $I_{45}$ | $I_{51}$ | $I_{52}$ | $I_{55}$ | $I_{61}$ | V |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X14Y | 1 | | | | | 1 | 1 | 1 | | | | 1 | 0 |
| X15Y | 1 | | | | | | | | 1 | 1 | 1 | 1 | 1 |
| X2Y | | 1 | 1 | | | | | | | | | 1 | 0 |
| X3Y | | | | 1 | 1 | | | | | | | 1 | 0 |
| Faults | | | | | | | | | 1 | 1 | 1 | | |

The symbolic notation $I_{jk}$ means execution of a statement that is on the rib $I_j$ and has index k. For instance, $I_{22}$ means execution of statement sequence of the rib $I_2$ at activation of the path X2Y and production operation that corresponds to the fragment of source program code:

else if ((x>=2) && (x<12)) f=2*x-3;

The diagnosis resolution for the test at the value of vector V = (0100) is determined by three possible faults: $F = I_{51}I_{52}I_{55}$. Value "1" of the vector V for a test-vector under consideration means that when issuing second pattern the activation of respective commands execution is took place. The minimal set of DNF terms, which make out all single faults of program fragments of a register transfer graph, is minimal diagnosis test. Next term set (here it coincide with complete test) makes out faults of all instructions, determined in DNF:

$$P = (111 \vee 141 \vee 151) \vee (111 \vee 121 \vee 151) \vee (21 \vee 31) \vee (11 \vee 12).$$

Reduction impossibility is conditional on that removal any term does not provide activation of one or several

fragments. Then complete and extended fault detection table is made that is formed by a term set above. Every obtained test pattern is divided on parts – terms. First test pattern $(111 \vee 141 \vee 151)$ consists of three terms: (111), (141) and (151). Every of them has own position in a column. All possible executable operations, which are designated $I_{jk}$, where j – rib identifier in a graph, k – statement that transforms data on j-th rib, is distinguished across. The graph path to which a term under consideration is applied is considered. For instance, term (141) is applied to first test pattern that activates the path X14Y. The extended fault detection table is:

| $T_i \backslash I_i$ | $I_{11}$ | $I_{22}$ | $I_{23}$ | $I_{31}$ | $I_{32}$ | $I_{41}$ | $I_{44}$ | $I_{45}$ | $I_{51}$ | $I_{52}$ | $I_{55}$ | $I_{61}$ | V |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $111_1$ | 1 | | | | | 1 | | | | | | 1 | 0 |
| $141$ | 1 | | | | | | 1 | | | | | 1 | 0 |
| $151_1$ | 1 | | | | | | | 1 | | | | 1 | 0 |
| $111_2$ | 1 | | | | | | | | 1 | | | 1 | 1 |
| $121$ | 1 | | | | | | | | | 1 | | 1 | 1 |
| $151_2$ | 1 | | | | | | | | | | 1 | 1 | 1 |
| $21_1$ | | 1 | | | | | | | | | | 1 | 0 |
| $31$ | | | 1 | | | | | | | | | 1 | 0 |
| $11$ | | | | 1 | | | | | | | | 1 | 0 |
| $21_2$ | | | | | 1 | | | | | | | 1 | 0 |

Every term number means execution of a statement on respective graph rib. First nimber "1" provides activation of the statement {1} $I_1$, so opposite respective column "1" is put. Column values of the extended fault detection table are moved from the FDT of code fragments that is defined on complete generalized test. But coordinate value is written for every test term. Extended fault detection table enable to show the results of every test pattern execution and to simplify the fault detection procedure with given resolution.

4. Diagnosis. In compliance with numbers of "1' in the output response vector V quantity of disjunctive CNF terms is formed. Every term is line-by-line writing of faults by logical operation "OR", which influence on distortion of output functional signals. Then transformation CNF to DNF by the Boolean algebra is carried out:

$F = (I_{11} \vee I_{51} \vee I_{61})(I_{11} \vee I_{52} \vee I_{61})(I_{11} \vee I_{55} \vee I_{61}) =$

$I_{11} \vee I_{11}I_{55} \vee I_{11}I_{61} \vee I_{11}I_{52} \vee I_{11}I_{52}I_{55} \vee I_{11}I_{52}I_{61} \vee I_{61}I_{11} \vee$

$\vee I_{11}I_{61}I_{51} \vee I_{11}I_{61} \vee I_{51}I_{11} \vee I_{11}I_{51}I_{55} \vee I_{11}I_{51}I_{61} \vee$

$\vee I_{11}I_{51}I_{52} \vee \vee I_{51}I_{52}I_{55} \vee I_{51}I_{52}I_{61} \vee I_{51}I_{61}I_{11} \vee$

$\vee I_{55}I_{61} \vee I_{51}I_{61} \vee I_{11}I_{61} \vee I_{11}I_{55}I_{61} \vee I_{11}I_{61} \vee$

$\vee I_{11}I_{52}I_{61} \vee I_{52}I_{55}I_{61} \vee I_{52}I_{61} \vee I_{61}I_{11} \vee I_{61}I_{51} \vee I_{61}.$

To reduce the obtained set of possible faults the Boolean algebra laws are used:

$A \wedge A = A; \; A \vee B = B \vee A; \; (A \vee B)C = AC \vee BC;$

$(A \vee B) \vee C = A \vee (B \vee C); \; A \vee A = A;$

$(A \wedge B) \vee A = A; \; (A \vee B) \wedge A = A$, it enables to obtain the expression:

$F = I_{11} \vee I_{11}I_{55} \vee I_{11}I_{61} \vee I_{11}I_{52} \vee I_{11}I_{52}I_{55} \vee$

$\vee I_{11}I_{52}I_{61} \vee I_{61}I_{11} \vee I_{11}I_{61}I_{51} \vee I_{11}I_{61} \vee$

$\vee I_{51}I_{11} \vee I_{11}I_{51}I_{55} \vee I_{11}I_{51}I_{61} \vee I_{11}I_{51}I_{52} \vee$

$\vee I_{51}I_{52}I_{55} \vee I_{51}I_{52}I_{61} \vee I_{51}I_{61}I_{11} \vee I_{55}I_{61} \vee$

$\vee I_{51}I_{61} \vee I_{11}I_{61} \vee I_{11}I_{55}I_{61} \vee I_{11}I_{61} \vee I_{11}I_{52}I_{61} \vee$

$\vee I_{52}I_{55}I_{61} \vee I_{52}I_{61} \vee I_{61}I_{11} \vee I_{61}I_{51} \vee I_{61} =$

$= I_{11} \vee I_{51}I_{52}I_{55} \vee I_{61}.$

Then such elements $I_{jk}$ from F, which are executed in other test patterns with value $V_i = 1$, are removed. A set of objects, contained the operations, which transform data at program execution uniquely and correctly, is formed:

$H = \{X14Y, X2Y, X3Y\} = \{(141) \vee (151) \vee (21_1) \vee (31) \vee$

$\vee (11) \vee (21_2)\} = I_{11} \vee I_{22} \vee I_{23} \vee I_{31} \vee I_{32} \vee I_{44} \vee I_{45} \vee I_{61}.$

After the reduction a single DNF term is obtained:

$F' = F \backslash H = (I_{11} \vee I_{51}I_{52}I_{55} \vee I_{61}) \backslash (I_{11} \vee I_{22} \vee$

$I_{23} \vee I_{31} \vee I_{32} \vee I_{44} \vee I_{45} \vee I_{61}) = I_{51}I_{52}I_{55}.$

It means that the software functions with error at execution one of the statements {1,2,5} on the rib $I_5$.

Really, an error takes place on linear program part that is applied to a rib of the statement sequence $I_5$, namely $I_{51}$ – execution of subtraction instead of summation.

More exact diagnosis (to within statement) is possible if to use the greater quantity of test points that complicates diagnosis because of necessity to make longer tests. The proposed method enables to analyze software on presence of errors in the code and helps to detect their location. Testing and verification of software is the main problem at programming, and its solving enables to raise software quality and to obviate unforeseen results of its execution. The proposed method is based on representation of software algorithm by the graph structure, where ribs are statement sequences or code fragments, and points are information monitoring points for making of assertions. Creation of minimal quantity of test patterns enables to decrease time of fault detection. At that tests have to cover all possible transactions. Test points quantity has to be minimal and sufficient for diagnosis of given resolution.

## IV. CONCLUSION

The innovative technologies of software and hardware testable design, based on effective test development and verification of digital system-on-a-chip components, are considered.

1. The general directions of utilization of the testable design technologies for digital systems-on-chips in the problems of software testing and verification are shown.

2. The universal model of software and hardware component in the form of directed register transfer and control graph, on which the testable design, test synthesis and analysis problems can be solved, is represented.

3. The metrics of hardware and software testability evaluation (controllability and observability), models of which are represented by directed register transfer and control graphs, is proposed.

4. The technology of software testing and diagnosis on basis of synthesis the graph register transfer models is proposed.

5. The practical importance of proposed methods and models is high interest of the software companies in innovative solutions of the effective software testing and verification problems above.

## REFERENCES

[1] Francisco DaSilva, Yervant Zorian, Lee Whetsel, Karim Arabi, Rohit Kapur, "Overview of the IEEE P1500 Standard", *ITC International Test Conference,* 2003, pp. 988–997.

[2] Abramovici M., Breuer M.A. and Friedman A.D, "Digital System Testing and Testable Design", *Computer Science Press*, 1998, 652 p.

[3] V.I.Hahanov, S.V.Chumachrnko, W.Gharibi, E.Litvinova, "Algebra-logical method for SoC embedded memory repair", *Proceedings of the 15 International Conference «Mixed design of integrated circuits and systems»*, Poland, 2008, pp. 481-486.

[4] Thatte S.M., Abraham J.A, "Test generation for microprocessors", *IEEE Trans. Comput.*, 1980, C-29, No 6, pp. 429-441.

[5] Sharshunov S.G, "Construction of microprocessor tests. 1. The general model. Data processing check", *Automation and telemechanics*, 1985, №11, pp. 145-155.

[6] Zorian Yervant, "What is Infrustructure IP?", *IEEE Design & Test of Computers*, 2002, pp. 5-7.

[7] Douglas Densmore, Roberto Passerone, Alberto Sangiovanni-Vincentelli, "A Platform-Based taxonomy for ESL design", *Design&Test of computers*, September-October, 2006, pp. 359-373.

[8] Bergeron, Janick, "Writing testbenches: functional verification of HDL models", *Boston: Kluwer Academic Publishers*, 2001, 354 p.

[9] Zorian Yervant, "Guest Editor's Introduction: Advances in Infrastructure IP", *IEEE Design and Test of Computers*, 2003, 49 p.

[10] Shameem Akhter, Jason Roberts, "Multi-Core Programming", *Intel Press*, 2006, 270 p.

**Vladimir Hahanov** – Dean of the Computer Engineering Faculty, Doctor of Science, Professor. IEEE Computer Society Golden Core Member.

1985 – Ph.D, "Digital systems models and testing methods for computer-aided design", Kharkov National University of Radio Electronics.

1996 – Dr. of Science, "Models and methods of digital microprocessors system for fault simulation and testing service", Kharkov National University of Radio Electronics.

2003 – till now dean of the Computer Engineering Faculty, Kharkov National University of Radio Electronics.

1997 – till now – professor of Kharkov National University of Radio Electronics. Kharkov Military University, Kharkov Academy of railway transport, Kharkov Academy of Culture, Kharkov Aerospace University.

1988 - 1997 - senior lecturer.

V. Hahanov has 510 publications, 7 books, and 2 patents.

Scientific work:

– Creation of the computer-aided system for logic simulation, test generation, faults diagnosis of digital devices;

– Systems and microprocessor-based structures;

– Two-framed cubic Algebra, cubic form of the graph representation, Cubic models of digital devices, deductive-parallel method of cubic fault simulation, topological deductive back traced parallel fault simulation method, cubic method of test generation;

– Algebra Logic fault localization and memory repair methods of SoC Functionality;

– Software tools (C++, Assembler, Fortran) for research.

High performance fault simulation and test generation development for complete digital systems and networks described hierarchical models. The development is based on Multi-Core architectures;

Design automation for testability with IEEE Boundary Scan standards and debugging tools for specialized microprocessor systems and digital systems processing;

Certification and verification of the hardware and software components of the computer systems and networks;

Design automation for educational applications in the field of computer engineering;

Digital Signal Processing and MPEG stsndards.

**Honors:**

1996 – "Best methodologist of University", Ukraine.

2000, 2001 – honors from "Best scientist of Kharkov region", Ukraine.

2003 – INTEL award of scientific projects competition.

2005 – The best professor of Ukraine.

2005 – Award from President of Ukraine.

2005 – IEEE Diploma for the IEEE conference organization.

2005 – IEEE Computer Society Golden Core Member.

2007 – IEEE Outstanding contribution Award.

Member of three specialized scientific boards for defense of thesis for a Doctor's degree: D 64.052.02 – systems of design automation, D 64.807.02 – information technologies in control systems.

Leader of the scientific seminar "Design automation and diagnosis of computational devices, systems and networks".

Chairman of the international symposium "IEEE East-West. Design and Test".

Member of 10 organization committee for the International Conferences

Member of IEEE Computer Society from 2000.

Member of High Examination Board of Ministry of Education of Ukraine.

Scientific supervisor of "Design & Test" R&D Lab.

Chief Scientist of Aldec Inc., cooperation with Cadence, Microsoft, Intel.

E-mail: hahanov@kture.kharkov.ua

**Eugenia Litvinova** – Assistent Professor, Doctor of philosophy. IEEE Society member.

1985 – Kharkov National University of Radioelectronics, speciality "Radioelectronic Designing and Production".

1996, Academic degree – candidate of technical science.

2001, Academic status – associate professor.

Senior Lector in Kharkov National University of Radio Electronics, Ukraine.

Over a period of time from 2000 year more 30 scientific publications were made.

Computer Engineering Faculty, Kharkov National University of Radioelectronics, Ukraine, Lenin Ave. 14, Kharkov, Ukraine, 61166, phone: (057) 70-21-421, (057) 70-21-326. E-mail: kiu@kture.kharkov.ua

**W. Gharibi** – PhD Student of the Kharkov National University of Radio Electronics, Computer Engineering Faculty.