KHARKOV NATIONAL UNIVERSITY OF RADIOELECTRONICS

Proceedings of IEEE East-West Design & Test Symposium (EWDTS'08)

SPONSORED BY

IEEE Computer Society Test Technology Technical Council

Lviv, Ukraine, October 9 – 12, 2008

CONTENTS

IEEE EAST-WEST DESIGN AND TEST SYMPOSIUM 2008
ORGANISING COMMITTEE

General Chairs
V. Hahanov – Ukraine
Y. Zorian – USA

General Vice-Chairs
M. Karavay – Russia
R. Ubar – Estonia

Program Chairs
S. Shoukourian – Armenia
D. Speranskiy – Russia

Program Vice-Chairs
M. Renovell - France
Z. Navabi - Iran

Publicity Chairs
C. Landrault - France
S. Mosin – Russia

Program Committee
E. J. Aas – Norway
J. Abraham – USA
A. Barkalov – Poland
R. Bazylevych – Ukraine
A. Drozd – Ukraine
E. Evdokimov – Ukraine
A. Chaterjee – USA
E. Gramatova – Slovakia
S. Hellebrand – Germany
A. Ivanov – Canada
V. Kharchenko – Ukraine
K. Kuchukjan – Armenia
A. Matrosova – Russia
V. Melikyan - Armenia
O. Novak – Czech Republic
A. Orailoglu – USA
Z. Peng – Sweden
A. Petrenko – Ukraine
P. Prinetto – Italy
J. Raik – Estonia
A. Romankevich – Ukraine
A. Ryjov – Russia
R. Seinauskas – Lithuania
S. Sharshunov – Russia
A. Singh – USA
J. Skobtsov – Ukraine
A. Stempkovsky – Russia
V. Tverdokhlebov – Russia
V. Vardanian - Armenia
V. Yarmolik – Byelorussia
A. Yessayan – Armenia

Steering Committee
M.Bondarenko – Ukraine
V. Hahanov – Ukraine
R. Ubar – Estonia
Y. Zorian – USA

Organizing Committee
S. Chumachenko – Ukraine
M. Kaminska – Ukraine
N. Kulbakova – Ukraine
M. Lobur – Ukraine
V. Obrizan – Ukraine
T. Sviridova – Ukraine

EWDTS CONTACT INFORMATION

Prof. Vladimir Hahanov
Design Automation Department
Kharkov National University of Radio Electronics,
14 Lenin ave,
Kharkov, 61166, Ukraine.

Tel.: +380 (57)-702-13-26
E-mail: hahanov@kture.kharkov.ua
Web: www.ewdtest.com/conf/

Testing Challenges of SOC Hardware-Software Components

Vladimir Hahanov, Volodimir Obrizan, Sergey Miroshnichenko, Alexander Gorobets
Computer Engineering Faculty, Kharkov National University of Radioelectronics, Lenin Ave. 14,
Kharkov, Ukraine, 61166, phone: (057) 70-21-421, (057) 70-21-326
E-mail: hahanov@kture.kharkov.ua; kiu@kture.kharkov.ua

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.

1. 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 to 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. 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.

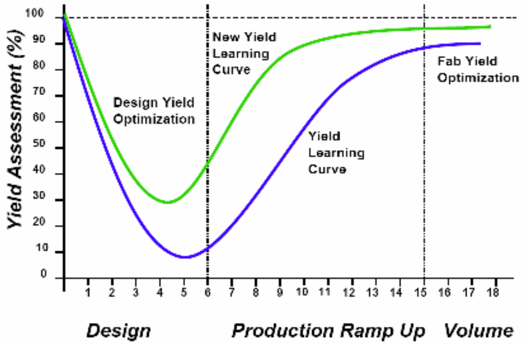

Fig. 1. Lifecycle curve of hardware-software complex

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.

2. 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 $R_i \to R_{k,}$ 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 $S_i \to S_k$ 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.

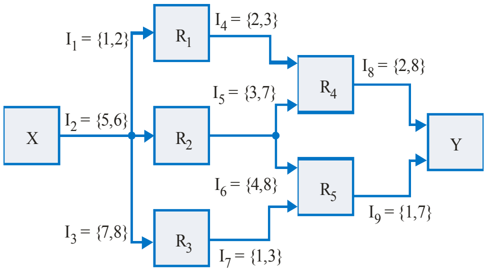

Fig. 2. Register transfer graph

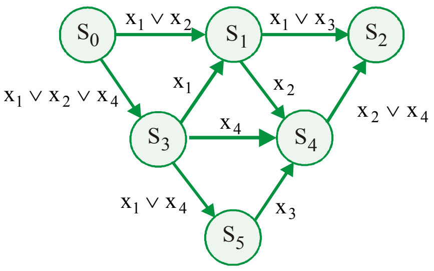

Fig. 3. Control automaton graph

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:

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

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

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

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

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

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 $I_{ij} \in I_i \approx (R_pR_q)$ that forms a command subset, attached to the rib (R_pR_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

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

$$=\frac{1}{k}\times\sum_{i=1}^{k}\left[\frac{d}{m}\right],$$

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_pR_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:

$$O(Y) = 1;$$

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

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

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

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

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

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

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

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:

$$O(S_2)=1;$$

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

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

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

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

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

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

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.

3. 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.

4. 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 .

[3] V.I.Hahanov, S.V.Chumachrnko, WGharibi, 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.

Camera-ready was prepared in Kharkov National University of Radio Electronics
by Dr. Svetlana Chumachenko and Volodymyr Obrizan
Lenin ave, 14, KNURE, Kharkov, 61166, Ukraine

Approved for publication: 20.09.2008. Format 60×841/8.
Relative printer’s sheets: . Circulation: 150 copies.
Published by SPD FL Stepanov V.V.
Ukraine, 61168, Kharkov, Ak. Pavlova st., 311

Матеріали симпозіуму «Схід-Захід Проектування та Діагностування – 2008»
Макет підготовлено у Харківському національному університеті радіоелектроніки
Редактори: Світлана Чумаченко та Володимир Обрізан
Пр. Леніна, 14, ХНУРЕ, Харків, 61166, Україна

Підписано до публікації: 20.09.2008. Формат 60×84$^1/_8$.
Умов. друк. арк. . Тираж: 150 прим.
Видано: СПД ФЛ Степанов В.В.
Вул. Ак. Павлова, 311, Харків, 61168, Україна