

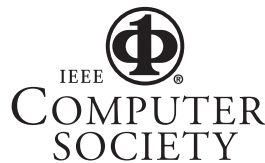
KHARKOV NATIONAL UNIVERSITY OF RADIOELECTRONICS

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

Copyright © 2008 by The Institute of Electrical and Electronics Engineers, Inc.

SPONSORED BY

IEEE Computer Society Test Technology Technical Council



Lviv, Ukraine, October 9 – 12, 2008

CONTENTS

A Systematic Approach for Evaluating Satellite Communications Systems Stefano Di Carlo, Paolo Prinetto, Alessandro Savino, Gabriele Tiotto, Paola Elia	13
Facilitating Testability of TLM FIFO: SystemC Implementations Homa Alemzadeh, Marco Cimei, Paolo Prinetto, Zainalabedin Navabi	17
A Model for Resistive Open Recursivity in CMOS Random Logic M. Renovell, M. Comte, N. Houarche, I. Polian, P. Engelke, B. Becker	21
An Optimized CLP-based Technique for Generating Propagation Sequences F. Fummi, V. Guarnieri, C. Marconcini, G. Pravadelli	25
Validation of a Mixed-Signal Board ATPG Method Val'erie-Anne Nicolas, Bertrand Gilles, Laurent Nana	29
A Low-Cost Optimal Time SICP air Generator I. Voyiatzis, H. Antonopoulou	35
Selected Cost Factors in Modeling and Testing Hardware and Semiconductor Defects by Dynamic Discrete Event Simulation Jack H. Arabian	41
HotSpot : Visualising Dynamic Power Consumption in RTL Designs T. English, K.L. Man, E. Popovici and M.P. Schellekens	45
Characterization of CMOS Sequential Standard Cells for Defect Based Voltage Testing A. Wielgus and W. A. Pleskacz	49
Testing the Control Part of Peripheral Interfaces S. Zielski, J. Sosnowski	55
Concurrent Processes Synchronisation in Statecharts for FPGA implementation Grzegorz Łabiak, Marian Adamski	59
Parallel Fault Simulation on Multi-core Processors Dmitry E. Ivanov	65
Synthesis of control unit with code sharing and chain modifications Alexander Barkalov, Larysa Titarenko, Jacek Bieganowski	68
FSMs Implementation into FPGAs with Multiple Encoding of States Arkadiusz Bukowiec, Alexander Barkalov and Larysa Titarenko	72
Reduction in the number of PAL macrocells for the effective Moore FSM implementation A. Barkalov, L. Titarenko, S. Chmielewski	76
Partial Reconfiguration of Compositional Microprogram Control Units implemented on an FPGA R. Wisniewski, Alexander A. Barkalov, Larysa Titarenko	80

Coverage-Directed Verification of Microprocessor Units Based on Cycle-Accurate Contract Specifications Alexander Kamkin	84
Code-Probability Entities for Constrained-Random Verification Diana Bodyan, Ghennady Bodyan	88
Multidimensional Loop Fusion for Low-Power Dmytro Lazorenko	92
A Synthesis of Common Models of Finite State Machines Using Input and Output Registers of Programmable Logic Devices Adam Klimowicz, Valeri Soloviev	96
An advanced Method for Synthesizing TLM2-based Interfaces Nadereh Hatami, Zainalabedin Navabi	104
Testing Combinational QCA Circuits Mehdi Azimipour	108
Dependability and Complexity Analysis of Inter-channel Connection Schemes for “N out of M” System-on-Chip Vyacheslav Kharchenko, Vladimir Sklyar, Georgiy Chertkov, Yuriy Alexeev, Ladislav Novy	113
Safety-Critical Software Independent Verification Based on Measurement of Invariants during Static Analysis Sergiyenko Volodymyr, Zavolodko Valeriy	117
Designing High Productivity Parallel Algorithms with Algebraic and Heuristic Programming Techniques Anatoliy Doroshenko, Mykola Kotyuk, Sergiy Nikolayev, Olena Yatsenko	121
Multiple Run Memory Testing for PSF Detection I. Mrozek , V.N. Yarmolik, E. Buslowska	125
The analysis of the start up control parameters of the asynchronous electric traction motors Gabriel Popa, Razvan A. Oprea, Sorin Arsene	131
A novel timing-driven placement algorithm using smooth timing analysis Andrey Ayupov, Leonid Kraginskiy	137
Digital Lock Detector for PLL Vazgen Melikyan, Aristakes Hovsepyan, Mkrtych Ishkhanyan, Tigran Hakobyan	141
Diagnosis of SoC Memory Faulty Cells for Embedded Repair Vladimir Hahanov, Eugenia Litvinova, Karina Krasnoyaruszhskaya, Sergey Galagan	143
Testing Challenges of SOC Hardware-Software Components Vladimir Hahanov, Volodimir Obrizan, Sergey Miroshnichenko, Alexander Gorobets	149

SoC Software Components Diagnosis Technology Svetlana Chumachenko, Wajeb Gharibi, Anna Hahanova, Aleksey Sushanov	155
Vector-Logical Diagnosis Method for SOC Functionalities Vladimir Hahanov, Olesya Guz, Natalya Kulbakova, Maxim Davydov	159
Testability analysis method for hardware and software based on assertion libraries Maryna Kaminska, Roman Prihodchenko, Artem Kubirya, Pavel Mocar	163
Different observation time strategies of outputs in diagnostics of sequential digital circuits Yu. A. Skobtsov, V. Yu. Skobtsov	168
Design and Implementation of a Parallel Adaptive Filter Using PBS-LMS Algorithm in a Convex Structure Ali Fathiyan, M. Eshghi	173
An IEEE 1500 Compatible Wrapper Architecture for Testing Cores at Transaction Level Fatemeh Refan, Paolo Prinetto, Zainalabedin Navabi	178
Power-Aware Embedded Software Design Fabian Vargas, Cláudia A. Rocha, Luís Fernando Cristófoli, Luciano Rocha	182
System Level Hardware Design and Simulation with System Ada Negin Mahani, Parnian Mokri, Zainalabedin Navabi	190
Automating Hardware/Software Partitioning Using Dependency Graph Somayyeh Malekshahi, Mahshid Sedghi, Zainalabedin Navabi	196
Reliable NoC Architecture Utilizing a Robust Rerouting Algorithm Armin Alaghi, Mahshid Sedghi, Naghmeh Karimi, Mahmood Fathy, Zainalabedin Navabi	200
Method for Modeling and Fault Simulation using Volterra kernels Pavlenko V., Fomin O.	204
Parity Prediction Method For On-Line Testing a Barrel-Shifter Drozd A., Antoshchuk S., Rucinski A., Martinuk A	208
RTL-TLM Equivalence Checking Based on Simulation Nicola Bombieri, Franco Fummi, Graziano Pravadelli	214
Estimation of the FPAA specification with use of the Artificial Neural Network Damian Grzechca, Tomasz Golonek	219
TUFFAN: A TLM Framework for Fast Architecture Exploration of Digital Systems Sheis Abolmaali, Parisa Razaghi and Zainalabedin Navabi	223
Code Optimization for Enhancing SystemC Simulation Time Homa Alemzadeh, Soheil Aminzadeh, Reihaneh Saberi, Zainalabedin Navabi	227

Automatic Test Pattern Generation Algorithm for Bridging Faults in Sequential Circuits F. Podyablonsky, N. Kascheev	231
Test Suite Consistency Verification Sergiy Boroday, Alexandre Petrenko, Andreas Ulrich	235
A 403-MHz Fully Differential Class-E Amplifier in 0.35 μm CMOS for ISM Band Applications Ghulam Mehdi, Naveed Ahsan, Amjad Altaf, Amir Eghbali	239
Signal Processing Verification System for the Programmable Digital Matched Filter Kharchenko H.V., Makovetskiy S.O., Tkalich I.O., Tsopa O.I., Vdovychenko Y.I	243
Building a Research Framework for Integrated Circuit Physical Design Andrey Kamaev, Kirill Korniyakov, Iosif Meyerov, Alexey Sidnev, Artem Zhivoderov	251
A High-speed and High Precision IDDQ Measurement for Consumer and Communication SoCs Yoshihiro Hashimoto, Yasuo Furukawa, Nguyen Ngoc Mai Khanh	254
Creating Test Environment for Consumer Video Devices Andrew Johnson, Oleksandr Yegorov	258
An Efficient Inner (De)Interleaver Architecture for DVB-T systems Mojtaba Rezayi, Mohammad Eshghi, and Hamid Reza Tanhaei	259
Redundant tests optimization Dmitriy Speranskiy, Ekaterina Ukolova	263
Sensor Web and Grid Technologies for Flood Applications N. Kussul, A. Shelestov, S. Skakun, Yu. Gripich	267
Persian Digit Recognition by Fourier Coefficients and Neural Networks Nasim Kazemifard, Pedram Azimi, Saeed Mozaffari	271
Deterministic Distinguishing Tests for Given Fault of Discrete Device Synthesis Dmitriy Speranskiy, Ivan Ukolov	276
Digital Implementation of General Regression Neural Network for Function Approximation Applications Saber Moradi, Mahmoud Tabandeh, Nasser Sadati	281
Hardware Implementation of Exponential Function Using a Mathematical approach Saber Moradi, Mahmud Tabandeh, Nasser Sadati	285
Automated Generation of Register Transfer Graph for Processors Victor Belkin	289
One Approach to Fault Dictionary Size Reduction Sergey Mironov, Dmitriy Speranskiy	295
Software engineering for recognition of electronic elements on the circuit board Dmitry Bagayev, Pavel Khrustalev	301

Automatic Identification of Radiotelephone Transmissions in the Maritime Communication Aleksandr V. Shishkin	306
QCA Parallel Prefix Adder Design S. Arab, H. Aghababa, B. Forouzandeh	310
Simple march tests for PSF detection in RAM Ireneusz Mrozek, Eugenia Buslowska	314
Improved Digital Signature Protocols On Elliptic And Hyperelliptic Curves Dolgov V.I., Nelasa G.V.	320
Cascade Structural Encoding of Binary Arrays Vladimir Barannik, Anna Hahanova	322
Mapping DSP Algorithms into FPGA Oleg Maslennikov, Anatolij Sergiyenko, Tatyana Lesyk	325
Precision of FTMPs reliability evaluation based on statistical experiments Romankevych A., Romankevych V., Chernyavskaya K	331
Discrete model for dynamics analysis of the nonlinear oscillating systems with long transient processes and complicated nature Zayats Vasyl	332
Deriving test suites for timed Finite State Machines M. Gromov, D. Popov, N. Yevtushenko	339
Checker Design for Arbitrary Subset of Unordered Code Words A. Matrosova, A. Malgin, N. Butorina	346
Multiple Stuck-at Fault and Path Delay Fault Testable Circuits A. Matrosova, V. Andreeva, A. Melnikov, E. Nikolaeva	356
Minimizing Path Length in Digital Circuits Based on Equation Solving N.Kushik, G.Sapunkov, S.Prokopenko, N.Yevtushenko	365
Utilizing HDL Simulation Engines for Accelerating Design and Test Processes Najmeh Farajipour, S. Behdad Hosseini and Zainalabedin Navabi	371
Performance evaluation of In-Circuit Testing on QCA based circuits Nasim Kazemifard, Maryam Ebrahimpour, Mostafa Rahimi, Mohammad Tehrani, Keivan Navi	375
Partitioning, Floor planning and detailed placement and routing techniques for schematic generation of analog netlist Bikram Garg, Rajeev Sehgal, Ashish Agrawal, Amarpal Singh, Manish Khanna	379
Parallel computer emulator for digital devices modeling Alexander Chemeris, Svetlana Reznikova	383

The Oscillations of an Overhead Contact Line Due to the Pantograph Raising R.A. Oprea, G.C.Popa, S.Arsene	387
Reverse Semantic Quality Control Methods in Software Engineering Vladimir L. Pavlov, Anatoliy Doroshenko, Konstantin Zhereb, Olexii Kuchaev	393
The Interplay of Reliability and Power Consumption in Design of SEU-Tolerant Latches for DSM Technology M. Fazeli, S. G. Miremdi, A. Patooghy	399
Evaluation of a Concurrent Error Detection Technique Using Power Supply Disturbance Fault Injection M. Fazeli, A. Patooghy and S.G. Miremadi	405
Embodying of High Performance Computation in Matlab Parallel Computing Toolbox for Detection of Spread Spectrum Signals Bohdan Yavorsky	411
Implementation of Finite State Machines on the Basis of an Embedded Memory Block V. Chapenko, K. Boule	414
On Macroplaces in Petri Nets Andrei Karatkevich	418
Testing of hardware and software for FPGA-based critical systems Yuliya Prokhorova, Sergey Ostroumov, Vladimir Sklyar	423
Luxury Wallet – new generation of the SoC based consumer products Mikhail Lodygin	427
Descriptor Neural Networks And Singular Implicit Dynamic Systems Rutkas A.A	429
Tools of the Computer Testing of Knowledge in Mathematical Disciplines Shkil A.S., Naprasnsk S.V., Tsimbaluyk E.S., Garkusha E.V.	431
Software for problem components estimation in photometric stereo reconstruction Bohdan Rusyn, Yuriy Lysak, Oleksiy Lutsyk	434
Method of Digital Treatment of the Information Received by Space Diversity Radars Dmitriy Vasiliev	436
Verification Challenges of Clock Domain Crossings D. Melnik, S. Zaychenko, O. Lukashenko.....	438
AUTHORS INDEX	441

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/

SoC Software Components Diagnosis Technology

Svetlana Chumachenko, Wajeb Gharibi, Anna Hahanova, Aleksey Sushanov
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

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.

1. Introduction

There are technologies of hardware testing and testable design, which enable to solve the problem of SoC service effectively. On the other hand, there are not effective models and methods of the given problem solving on the electronic technology market.

To realize testable design and diagnosis of SoC software components the universal model of software components representation in the form of register transfer and control graph is developed. An algorithm of software diagnosis is proposed. An instance of software diagnosis technology utilization is considered.

The research aim is adaptation of the hardware testing methods to the service of SoC software components.

The research problems: 1) Adaptation of Thatte-Abraham-Sharshunov register transfer model to the solving of software testing problem; 2) Application of the model for faulty statements diagnosis on basis of use the fault detection table.

2. 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:

$$S = (x) + \omega(x),$$
$$x = \begin{cases} x + 3; & x < 2; \\ 2x - 3; & 2 \leq x < 12; \\ -3x + 7; & x \geq 12; \end{cases}$$

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

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

Listing 3.1.

```
#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

```
else w=sin(Pi*x)+2;
the following one is written:
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. 1); graph points are points of information monitoring (registers, variables, memory), which are used for forming of assertions too.

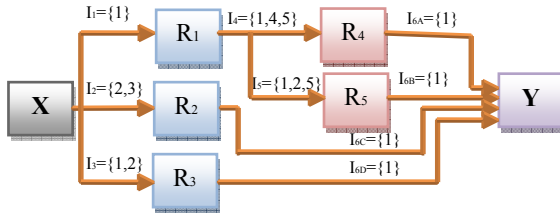


Fig. 1. Register transfer graph

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. 1) 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:

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

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:

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

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:

$T_i \backslash I_j$	I11	I22	I23	I31	I32	I41	I44	I45	I51	I52	I55	I61	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:

$T_i \backslash I_j$	I11	I22	I23	I31	I32	I41	I44	I45	I51	I52	I55	I61	V
111 ₁	1					1						1	0
141 ₁	1						1					1	0
151 ₁	1							1				1	0
111 ₂	1								1			1	1
121 ₁	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 number “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:

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

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

$$A \wedge A = A; A \vee B = B \vee A; (A \vee B)C = AC \vee BC; (A \vee B) \vee C = A \vee (B \vee C); A \vee A = A; (A \wedge B) \vee A = A; (A \vee B) \wedge A = A, \text{ it enables to obtain the expression:}$$

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

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:

$$H = \{X14Y, X2Y, X3Y\} = \{(141) \vee (151) \vee (21_1) \vee (31) \vee (11) \vee (21_2)\} = I_{11} \vee I_{22} \vee I_{23} \vee I_{31} \vee I_{32} \vee I_{44} \vee I_{45} \vee I_{61}$$

After the reduction a single DNF term is obtained:

$$F' = F \setminus H = (I_{11} \vee I_{51} I_{52} I_{55} \vee I_{61}) \setminus (I_{11} \vee I_{22} \vee I_{23} \vee I_{31} \vee I_{32} \vee I_{44} \vee I_{45} \vee I_{61}) = I_{51} I_{52} I_{55}.$$

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.

3. Conclusion

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

1. 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.

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

3. 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, 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 Infrastructure 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 симпозиуму «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 60×84¹/₈.

Умов. друк. арк. . Тираж: 150 прим.

Видано: СПД ФЛ Степанов В.В.

Вул. Ак. Павлова, 311, Харків, 61168, Україна