ISSN 1392 – 124X INFORMACINĖS TECHNOLOGIJOS IR VALDYMAS, 2004, Nr. 3(32)

REALIZATION-INDEPENDENT TESTING OF SEQUENTIAL CIRCUITS. EXPERIMENTAL RESULTS

Eduardas Bareiša, Vacius Jusas, Kęstutis Motiejūnas, Rimantas Šeinauskas

Software Engineering Department, Kaunas University of Technology
Studentų St., LT-3031 Kaunas, Lithuania

Abstract. In this paper we analyze the situation when the tests are generated for a particular implementation. In this case there rises a question – can a test generated for one implementation be used for another implementation? Naturally, that a test generated according to one structure may not detect all specified faults of another structure. In this work we explore the test quality of one realization for detecting faults of other realizations. We analyze only such implementations that are generated by the same synthesis tool according to the same description, changing only the target library used during the synthesis. We have performed various experiments with sequential benchmark circuits. Our experiments show that the fault coverage surprisingly coincides. We think that there are two possible explanations of the outcome of the experiments. Firstly, the test redundancy is very high., Secondly, the combinational parts of all analyzed sequential circuits have simple logic and, therefore, the test sequences generated for particular realization are equally good for other realizations as well.

1. Indroduction

Many recent system-on-a-chip (SOC) ICs incorporate pre-designed and reusable circuits, variously referred to as intellectual property (IP) circuits or cores. Such circuits are frequently supplied by third-party vendors and are extremely hard to test when embedded in an SOC because their functions are specified only in high-level terms. This is done either to protect the circuits' IP content or else to allow system designers to synthesize their own low-level (gate-level) implementations. The tests can be generated for a high level description in order to reuse them for all possible implementations [1]. However, such tests usually can not guarantee detection of all specified faults in all possible implementations. Consequently, if we consider realization-independent testing, we can only speak about such realizations that fulfill specific requirements or have a particular structure [2, 3].

In this paper we analyze the situation when the tests are generated for a particular implementation. In this case there naturally rises a question – can a test generated for one implementation be used for another implementation? The same core can have distinct descriptions; e. g. a parallel or sequential carry can be realized in an adder. Naturally, that a test generated according to one structure may not detect all specified faults of another structure. This case is studied in [4, 5], where it is shown that the deviation of the stuck-at fault coverage in combinational circuits can be up to 18% high [4]. The employment of different synthesis tools can have an influence on the test quality as well.

In [4] H.Kim and J.P.Hayes synthesized two different gate-level implementations of the combinational example circuits, one for low area and another for high speed. The stuck-at fault tests for the gate-level designs were generated using the conventional ATPG program Atalanta [6]. It is stated that Atalanta tests provide 100% stuck-at fault coverage only for the gate-level designs at which they were targeted and fairly poor coverage for the others. The most impressive number is provided for the ISCAS'85 benchmark circuit c880, namely, 100 % stuck-at fault test for high speed realization detects only 82.2% stuck-at faults of the low area realization. Similar experiments are described in [5], too. In [5] it is reported that for the ISCAS'85 benchmark circuit c880 99.8 % stuck-at fault test for high speed realization detects already 99.7% stuck-at faults of the low area realization.

In [7] various experiments with ISCAS'85 combinational benchmark circuits are performed. The results of experiments show that the test sets generated for a particular circuit realization fail to detect in average only less than one and a half percent of the stuck-at faults of the re-synthesized circuit but in some cases this figure is more than nine percent.

The possibilities of supplementing or expanding a particular realization test having a purpose to enhance test quality for detecting of various defects are analyzed in [7-11]. The defect coverage that can be achieved with test sets for stuck-at faults may not be

sufficient. In order to increase the defect coverage of a test set for stuck-at faults, in [8] and [9] *n*-detection test sets were considered. An *n*-detection stuck-at test set is one where each stuck-at fault *f* is detected by *n* different input patterns, or by the maximum number of input patterns if *f* has fewer than *n* different input patterns that detect it. Experiments with *n*-detection stuck-at test sets reported in [8] and [9] show that it is possible to enhance the defect coverage using this approach. In various types of experiments performed in [10] and [11] *n*-detection test sets were shown to be useful in achieving a high defect coverage for all types of circuits and for different fault models. Similar results for double test sets are obtained in [7], too. Another interesting outcome of the experiments performed in [7] is that the supplement of the test set with sensitive adjacent test patterns significantly increases the fault coverage and is a very cheap way to adopt test patterns for the re-synthesized gate level description of the IP core.

All results described in [4-11] concern only combinational circuits and there are no publications concerning sequential circuits. In this work we will explore the test quality of one realization for detecting faults of other realizations regarding sequential circuits. We will analyze only such implementations that are generated by the same synthesis tool according to the same description, changing only the target library used during the synthesis. The ITC'99 sequential benchmark circuits have been selected for experiments.

The structure of the paper is as follows. We analyze the parameters of ITC'99 benchmark circuits in Section 2. We present the experimental results in Section 3. We finish with conclusions in Section 4.

2. The parameters of considered circuits

As it was already mentioned in the introduction the core can be synthesized by different electronic design automation systems and mapped into different cell libraries and manufacturing technologies. In [7] we have presented the experimental results that show how the test set of the core covers the faults of new implementations for combinational circuits. In this paper we are going to consider the same problem for sequential circuits. The original ITC'99 benchmark circuits [12] were chosen for experiments. The combinational part of these circuits has been re-synthesized with the Synopsys Design Compiler program by the default mode and by using an AND-NOT cell library of two inputs. The three realizations have been analyzed:

R1 – the non-redundant benchmark circuit

R2 – Synopsys Design Optimization, the target library – class.db (default mode)

R3 - Synopsys Design Optimization, the target library – and_or.db

The parameters of the original ITC'99 benchmark circuits are given in Table 1 and Figures 1, 2. The columns are denoted as follows: Gates – the number of gates, FF – the number of flip-flops, PI – the number of primary inputs, PO – the number of primary outputs, Best fault coverage % - the best published in the papers stuck-at fault coverage of the original ITC'99 benchmark circuits reached using test generators Hitec, RAGE, TetraMAX or GATO, R1, R2, R3– the number of stuck-at faults in the circuit realizations R1, R2, R3 respectively, Δ - the difference between the maximum and the minimum stuck-at faults numbers, % - the percent of the difference to the maximum stuck-at faults number.

Table 1. The parameters of the original ITC'99 benchmark circuits

Circuits	Gates	FF	PI	PO	Best fault coverage %	Number of stuck-at faults			Δ	%
						R1	R2	R3		
b01	40	5	4	2	100.00	268	246	278	32	12
b02	18	4	3	1	99.22	128	126	128	2	2
b03	111	30	6	4	73.24	822	782	784	40	5
b04	394	66	13	8	89.58	2640	2614	2666	52	2
b05	570	34	3	36	40.00	3362	2880	3132	482	14
b06	48	9	4	6	94.15	346	334	336	12	3
b07	321	51	3	8	50.00	2198	2302	2488	290	12
b08	154	21	11	4	88.10	800	812	828	28	3
b09	100	28	3	1	87.23	736	758	772	36	5
b10	137	17	13	6	93.59	952	964	1078	126	12
Total						12252	11818	12490		

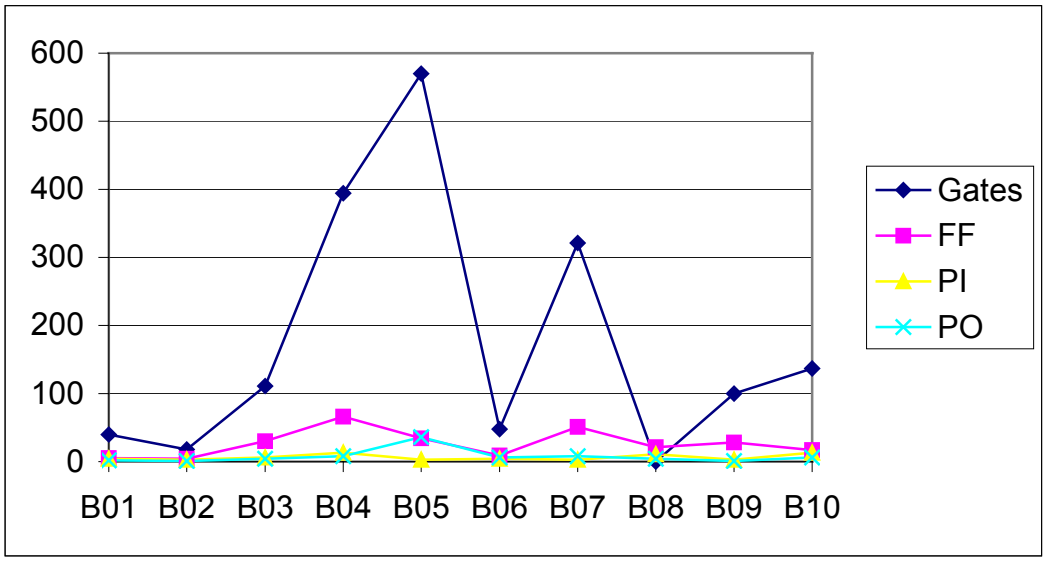

Figure 1. The parameters of the original ITC'99 benchmark circuits

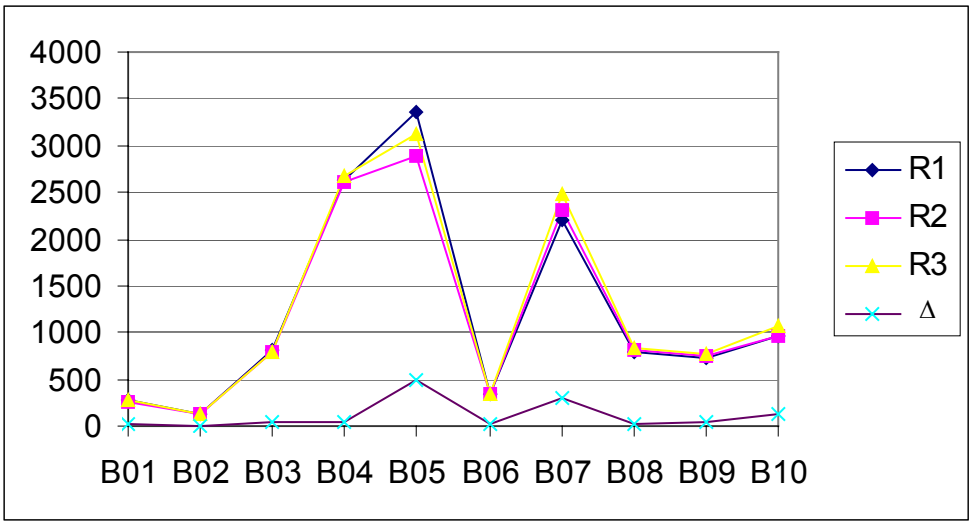

Figure 2. The number of stuck-at faults for each realization

We can see the number of stuck-at faults for each realization in Table 1 and Figure 2. The benchmark circuit realizations using the target library and_or.db have more stuck-at faults in total. The most optimized are realizations using the target library class.db. But the percent of the difference between the maximum and the minimum numbers to the maximum number of stuck-at faults, which varies from 2 to 14, is not so big as for analyzed combinational ISCAS'85 benchmark circuits (from 8 to 53 percent).

3. Experimental results

Most sequential ATPG algorithms are the direct extensions of combinational algorithms applied to the iterative logic array model [13] of the circuit under test. An advanced circuit description language like VHDL gives an opportunity to apply an iterative logic array model of a sequential circuit, and in the case of the use of the combinational ATPG to manage the search space flexibly.

A test generation approach for sequential circuits based on the iterative logic array model is described in [13]. We will shortly remind the main features of this approach. Each component of the state vector can have one of five values, namely {0, 1, X, D, notD}. If a test exists, 0 or 1 can replace the X value, hence only four values need to be considered. It is clear that in testing a circuit it is never necessary to enter some state twice, therefore each state vector can be restricted to be unique, and there are 4^n such unique states, where n is the number of state variables. The test generation procedure given in [13] consists of the following three steps:

1. Set k to 1 for the number of copies (combinational cells) of the iterative circuit model. Set

don't-care value X for all state variables of the first copy.

2. Construct the k-iterative array model of the circuit.
3. Apply the combinational ATPG for a k-iterative array model of the circuit. If no test vectors are found for undetected faults, increment k by one and repeat (2). Terminate when $k>4^n$. If no test vectors are found for some faults, the circuit is redundant.

We implemented the procedure described above by means of the SYNOPSYS combinational ATPG for stuck-at faults. Of course, the extent of the state search space 4^n is purely theoretical and not applicable for real circuits. In our experiments we stopped incrementing the length k of the iterative logic array model when the appliance of the last twenty additional combinational copies of the circuit did not increase the fault coverage or in the case when the SYNOPSYS combinational test generator was not able to deal with such enlarged circuit.

The combinational ATPG (SYNOPSYS) generates test vectors of the length k*PI, where k – the number of combinational copies, PI – the number of primary inputs in one copy. In order to apply these test vectors to initial circuits they were folded in test sequences of k test vectors of length equal to PI each. The test sets have been generated for each original ITC'99 benchmark circuit and than reused for two other realizations. A Sun Ultra 5 computer was used for the test generation. The results of the experiment are presented in Table 2.

Table 2. The iterative logic array model. Undetected stuck-at faults for the three realizations

Circuit	Nr.	Sequence length	Generation time	R1				R2				R3			
				%	D	U	S	%	D	U	S	%	D	U	S
b01	18	12	1 sec.	100	268	0	8	100	246	0	8	100	278	0	7
b02	10	11	<1 sec.	99.2	127	1	6	97.6	123	3	5	98.4	126	2	6
b03	33	16	7 sec.	74,8	615	207	13	72,0	563	219	12	73,9	579	205	5
b04	74	15	54 sec.	90.7	2395	245	35	90,8	2374	240	37	91,0	2425	241	36
b05	55	257	500 sec.	50,8	1708	1654	3	56,5	1627	1253	3	55,5	1739	1393	3
b06	16	11	<1 sec.	93.5	290	20	7	93,4	312	22	8	94,3	317	19	8
b07	34	289	6000 sec.	75,0	1648	550	2	72,5	1670	632	2	73,0	1816	672	2
b08	48	65	<400 sec.	98.2	786	14	23	98,6	801	11	20	98,6	816	12	20
b09	37	81	328 sec.	87.9	647	89	12	86,3	654	104	13	84,5	652	120	13
b10	39	25	377 sec.	93.6	891	61	17	93,5	901	63	18	94,3	1017	61	18

R1 – the original ITC'99 benchmark circuit

R2 –Synopsys Design Optimization, the target library – class.db

R3 – Synopsys Design Optimization, the target library – and_or.db

Nr. – the number of test sequences

% – the fault coverage

D – the number of detected faults

U – the number of undetected faults

S – the number of test sequences that add their value to the fault coverage

The second test generation approach used in our experiments is based on the algorithmic level of the circuit description. The basic ideas presented in [14], where we consider input – output paths testing in combinational circuits, were employed for sequential circuits as well. In this section we present only additional information concerning sequential circuits.

The algorithmic description is accomplished in some high-level programming language, for example, C. It excludes clock and reset information. Such a model expresses only a function carried out by a circuit. We call such a model of the circuit a black box model. In order to generate tests based on the black box model, a fault simulation is used. The tests are generated for PP faults [14].

As we have mentioned, a model of the circuit at the algorithmic level has no clock information. Therefore, generated test patterns based on such model may be applied directly only if a circuit is combinational. If a circuit is sequential, these test patterns require an additional treatment. For this purpose the test frames are used.

A sequential circuit has a defined clock and control signal sequence in order to carry out some operations with data. Such a control and clock input stimuli sequence defines the mode of data transactions. We call a sequence of the input stimuli, which defines the values of a signal only for some inputs and leaves undefined values for other inputs, a *test frame*. A sequential circuit together with a test frame can be presented as a function whose outputs depend only on inputs and do not depend on the inner state of a

circuit; this means that the sequential circuit is like the combinational circuit. A test frame defines those inputs which control the storage of sequential circuit and leaves undefined inputs, which are devoted for data. A test frame is defined before the test generation. A constructed model of a circuit is very closely related to the test frame. This model is created according to a test frame. A test frame defines a mode of transferring of test cases that it gets from a test generation to the test sequences. A test frame is filled up with the values of test cases [14].

As it was mentioned above ITC"99 benchmark circuits B01 – B10 [12] were chosen for the experiments. For these circuits there were written the models in the programming language C according to their VHDL models. In the C models clock and reset signals were eliminated. Only the model of the circuit b08 was changed by a combinational equivalent, but nevertheless the test frame was used for this circuit too. Some VHDL circuits like b05 have several parallel processes, but C is a sequential language. Therefore, there was a challenge to solve the problem of parallel processes using C programming language. There is no common recipe for this problem. In every case an individual approach was used. For example, the parallel processes of the circuit b05 are activated by different signals. Therefore, the main process, which is sensible to clock and reset signals, was found. All other processes were accomplished as subprograms, which are called when the active signal has changed its value. The structure of test frames for all circuits was very simple – every test sequence began with Reset and Clock and the following test vectors of the sequence had only Clock.

The test sets have been generated for PP faults and then analyzed how well they cover stuck-at faults in all three realizations. A Pentium 4.3 GHz Hyper-Threading PC was used for test generation. The results of the experiment are presented in Table 3.

Table 3. The black box model. Undetected stuck-at faults for the three realizations

Circuit	Nr.	Sequence length	Generation time	R1				R2				R3			
				%	D	U	S	%	D	U	S	%	D	U	S
b01	65	12	1 sec.	100	268	0	8	100	246	0	7	100	278	0	8
b02	28	11	1 sec.	99,22	127	1	4	99,22	125	1	4	99,22	127	1	4
b03	384	16	1 sec.	74,82	615	207	12	72	563	219	11	73,9	579	205	10
b04	890	15	2 sec.	90,7	2395	245	37	90,8	2374	240	38	91	2425	241	36
b05	24	257	1 sec.	50,8	1708	1654	2	56,5	1627	1253	3	55,5	1739	1393	3
b06	187	11	1 sec.	93,5	290	20	6	93,4	312	22	6	94,3	317	19	6
b07	36	289	1 sec.	75	1648	550	1	72,5	1670	632	1	73	1816	672	1
b08	274	65	3 sec.	98,2	786	14	20	98,6	801	11	20	98,6	816	12	19
b09	1228	81	2 sec.	87,9	647	89	16	88,1	668	90	23	88,1	680	92	26
b10	1225	25	3 sec.	93,59	891	61	22	93,5	901	63	21	94,3	1017	61	22

R1 – the original ITC'99 benchmark circuit

R2 –Synopsys Design Optimization, the target library – class.db

R3 – Synopsys Design Optimization, the target library – and_or.db

Nr. – the number of test sequences

% – the fault coverage

D – the number of detected faults

U – the number of undetected faults

S – the number of test sequences that add their value to the fault coverage

We cannot examine the results of experiments with sequential circuits in the same way as we examined the combinational circuits [7]. The reason is that for all these circuits except b01 we do not have 100% fault coverage and, therefore, we do not know the number of redundant faults in separate realization.

However, if we examine Tables 2 and 3 together we can see that the fault coverage for all realizations in both tables surprisingly coincides. There are only two exceptions – the fault coverage values in the tables differ for realizations R2 and R3 of the circuits b02 and b09. Afterwards we additionally conducted the third experiment using adjacent input vectors [7] for the black box model approach. The appliance of adjacent input vectors for combinational circuits was very effective [7]. However, it was not the same case for sequential circuits. We got exactly the same results as those presented in Table 3. Therefore, in our opinion, there is a very high probability that the obtained stuck-at fault coverage is maximal in every particular realization for the used number of copies.

We also considered another possible explanation of such outcome of the experiments. Namely, if we examine the quantities of generated test sequences (column "Nr." in both tables) and the quantities of test sequences that add their value to the fault coverage

(columns "S"), we can see that the test redundancy is very high. It ranges from 211% (circuit b04) to 1833% (circuit b05) for the iterative logic array model approach and from 700% (circuit b02) to 7675% (circuit b09) for the black box model approach. By reason of such a high redundancy the test sequences generated for particular realization may be equally good for other realizations as well. That's why we performed the fourth experiment. The test sets generated for each original ITC'99 benchmark circuit using the iterative logic array model approach were minimized and then the minimized tests were reused for two other realizations. The results of the experiment are presented in Table 4.

Table 4. The iterative logic array model. Minimized tests. Undetected stuck-at faults for the three realizations

Circuit	Nr.	R1				R2				R3			
		%	D	U	S	%	D	U	S	%	D	U	S
b01	18	100	268	0	8	100	246	0		100	278	0	
b02	10	99.2	127	1	6	97,6	123	3		98,4	126	2	
b03	33	74,8	615	207	13	72,0	563	219		73,9	579	205	
b04	74	90.7	2395	245	35	90,8	2374	240		91,0	2425	241	
b05	55	50,8	1708	1654	3	56,5	1627	1253		55,5	1739	1393	
b06	16	93.5	290	20	7	93,4	312	22		94,3	317	19	
b07	34	75,0	1648	550	2	72,5	1670	632		73,0	1816	672	
b08	48	98.2	786	14	23	98,6	801	11		98,6	816	12	
b09	37	87.9	647	89	12	86,0	652	106		83,9	648	124	
b10	39	93.6	891	61	17	93,5	901	63		94,3	1017	61	

R1 – the original ITC'99 benchmark circuit

R2 –Synopsys Design Optimization, the target library – class.db

R3 – Synopsys Design Optimization, the target library – and_or.db

Nr. – the number of test sequences

% – the fault coverage

D – the number of detected faults

U – the number of undetected faults

S – the number of test sequences that add their value to the fault coverage

If we compare the results in Tables 2 and 4 we can see that minimized tests for most analyzed circuits give the same stuck-at fault coverage as well. Only for one circuit b09 the fault coverage is lower than maximal in a particular realization; for the realization R2 the difference is 0.3% and for the realization R3 - 0.6%. We think that there is a second possible explanation of the results of all our experiments with sequential circuits. Namely, the combinational parts of all analyzed sequential circuits have simple logic and, therefore, the tests generated for a particular realization are equally good for other circuit realizations in almost all cases.

4. Conclusions

We cannot examine the results of experiments with sequential circuits in the same way as with combinational circuits. The reason is that for almost all these circuits we do not have 100% fault coverage and, therefore, we do not know the number of redundant faults in separate realizations.

However, if we examine all our experimental results we can see that the fault coverage surprisingly coincides. In our opinion there is a very high probability that the obtained stuck-at fault coverage is maximal in every particular realization for the used number of copies. We also considered another possible explanation of such outcome of the experiments. Namely, if we examine the quantities of generated test sequences and the quantities of test sequences that add their value to the fault coverage we can see that the test redundancy is very high. It ranges from 211% 1833% for the iterative logic array model approach and from 700% to 7675% for the black box model approach. By reason of such a high redundancy the test sequences generated for a particular realization may be equally good for other realizations as well. This explanation sustains and the fact that he appliance of adjacent input vectors, which was very effective for combinational circuits, did not improve the test quality for considered sequential circuits.

We think that there is a second possible explanation of the results of our experiments with sequential circuits. Namely, the combinational parts of all analyzed sequential circuits have simple logic and, therefore, the tests generated for a particular realization are equally good for other circuit realizations in almost all cases. The impact of the complexity of combinational parts on the fault coverage can be investigated in near future.

References

[1] **Y. Zorian, S. Dey, M. Rodgers.** Test of Future System-on-Chips. *Proceedings of the* 2000 *International Conference on Computer-Aided Design, November*, 2000, 392-398.

[2] **S. B. Akers.** Universal Test Sets for Logic Networks. *IEEE trans. Computers*, *Vol*. C-22, 1973, 835-839.

[3] **R. Betancourt.** Derivation of Minimum test sets for Unate Logic Circuits. *IEEE Trans. Computers, Vol*. C-20, *Nov*. 1971, 264-1269.

[4] **H. Kim, J.P. Hayes.** High-Coverage ATPG for datapath circuits with unimplemented blocks. *Proc. Int. Test Conf., Oct*. 1998, 577-586.

[5] **J. Yi, J.P. Hayes.** A Fault Model for Function and Delay Testing. *Proc. of the IEEE European Test Workshop, ETW*'01, 2001, 27-34.

[6] **H.K. Lee, D.S. Ha.** On the generation of test patterns for combinational circuits. *Technical Report, Department of Electrical Eng., Virginia Polytechnic Institute and State University*, 1993, 12-93.

[7] **E. Bareisa, V. Jusas, K. Motiejunas, R. Seinauskas.** The Influence of Circuit Re-synthesizing on the Fault Coverage. *Information technology and control, Kaunas, Technologija*, 2004, *No*.2(31).

[8] **S.C. Ma, P. Franco, E.J. McCluskey.** An experimental chip to evaluate test teshniques. *Experimental results, in Proc.* 1995 *Intl. Test Conf., Oct* 1995, 663-672.

[9] **S.M. Reddy, I. Pomeranz, S. Kajihara.** On the Effects of Test Compaction on Defect Coverage. *Proc. VLSI Test Symp., Apr*. 1996, 430-435.

[10] **I. Pomeranz, S.M. Reddy.** On n-Detection Test Sets and Variable n-Detection Test Sets for Transition Faults. *Proc.* 17*th VLSI test Symp.*, April 1999, 173-179.

[11] **H. Takahashi, K.K. Sulaja, Y. Takamatsu.** An Alternative Method of Generating Tests for Path Delay Faults Using N -Detection Test Sets. *Proc. Of the* 2002 *Pacific Rim International Symposium on Dependable Computing* (*PRDC*'02), 2002.

[12] **F. Corno, M.Sonza Reorda, G. Squillero.** RT-level ITC99 Benchmarks and First ATPG Results. *IEEE Design & Test of Computers, July-August* 2000, 44-53.

[13] **M.A. Breuer, A. D. Friedman.** Diagnosis & Reliable Design of Digital Systems. *Computer Science Press*, 1976.

[14] **V.Jusas, R.Seinauskas.** Automatic Test Patterns Generation forSimulation-based Validation. *Proc. of the 8-th Biennal Baltic ElectronicsConference. ISBN* 9985-59-292-1. *Tallinn Technical University, October* 6-9, 2002, *Tallinn, Estonia*, 295-299.