ISSN 1392 - 1215 ELEKTRONIKA IR ELEKTROTECHNIKA. 2004. Nr.7 (56)

T 170 ELEKTRONIKA

Testing of FPGA Logic Cells

E. Bareiša, V. Jusas, K. Motiejūnas, R. Šeinauskas

Software Engineering Department, Kaunas University of Technology
Studentų St. 50-406, Kaunas, Lithuania, e-mail: eduardas.bareisa@ktu.lt

1. Introduction

Field programmable gate arrays (FPGAs) are digital devices that can implement logic circuits required by users in the field. As a result, most prototypes and many production designs are now implemented on FPGAs, making hardware implementation economically feasible even for those applications which were previously restricted to software implementation. There are many different architectures of FPGAs driven by different programming technologies. One important class is the SRAM based FPGAs, also called the look-up table (LUT) FPGAs. Such a programmable circuit consists of a matrix of logic modules and interconnection elements.

As the use of FPGAs in commercial products becomes more widespread, the importance of reliability and test obtains a great value. For reprogrammable FPGAs two types of testing can be considered [1]. One is the testing of unprogrammed FPGAs which is accomplished by the producer right after manufacturing (Manufacturing-Oriented Test Procedure, MOTP). The other is the testing of the programmed FPGA which is accomplished by the user when the device is deployed by a given application (Application-Oriented Test Procedure, AOTP). An unprogrammed FPGA can realize many different programmed FPGAs by loading different programs. Theoretically, to test the unprogrammed FPGA, we might have to test all the programmed FPGAs obtained from the unprogrammed FPGA. FPGAs appear as very complex circuits and all papers that consider the testing of FPGA use a classical divide and conquer approach. Usually each paper targets a specific FPGA part: the logic cells [2, 3], the memory cells [4], the interconnect cells [5]. All above mentioned papers are devoted to unprogrammed FPGAs. There are only few papers devoted to programmed FPGAs [1, 6, 7]. These papers also use the same divide and conquer approach and consider faults only in the logic cells. It is possible to distinguish among them only two slightly different approaches. Both of them agree that the test vectors computed by a gate level test pattern generator with the gate level circuit netlist and stuck-at faults produce a low coverage according to the FPGA implementation. A FPGA logic implementation can be used for the test pattern generation and leads to better results. The papers [1, 6] consider the adoption of the classical test pattern generator investigating the active logic cell FPGA description of the application configured in FPGA. But a FPGA description is by definition much more complex than the circuit netlist because of the inherit flexibility of the FPGA. Consequently, the authors [1, 6] noticed that the initial FPGA description has a huge number of application configuration (AC) redundant faults. Therefore they defined different classes of AC-redundant faults: a) AC-redundant faults due to logical redundancy; b) AC-redundant faults due to unused logic; c) AC-redundant faults due to the constant signal. Elimination of redundant faults from the list gives the reduction from 4% to 93%. But in that approach, the model describing the possible faults affecting the configuration memory is approximate, since it does not consider the faults affecting the values of the memory cells composing each LUT. It considers only the faults affecting the value of the output of the LUT.

On the other hand, the approach [7] extends the set of faults in comparing with [1, 6] by adding the faults affecting the LUT bit cells. But the paper [7] does not propose the method how to generate the test patterns to detect these faults. The paper only states the fact that the fault coverage is generally low of the faults which affect the LUT bit cells when the classical test pattern generator is used. The paper also demonstrates the fact that there is almost no difference which test pattern generator to use: the gate-level commercial ATPG or a RT-level academic ATPG.

In this paper, we propose an approach of the exhaustive testing of logic cells for an FPGA configured circuit to implement a given application. In our approach, we apply the exhaustive test pattern generation for every logic cell. Such an approach lets to neglect the inner structure of the logic cell and test patterns generated according to our approach are able to detect all inner defects of logic cells that could be detected by a single test pattern. As experimental results show, the exhaustive testing cannot be established for all logic cells due to problems controllability or observability. Reconfiguring an application-oriented FPGA into a tree like structure can solve this problem, but the FPGA configuration process is an excessively time-consuming.

The paper is organized as follows. Section 2 introduces a new model of the configurable logic block.

Section 3 analyzes the process of the test pattern generation for the FPGA configured circuit. Section 4 presents and comments experimental results. Section 5 draws main conclusions.

2. The circuit transformation

In AOTP approach the user needs to test only the part of the FPGA used by the configured application. We restrict our investigation to CLBs only. But as distinct from the approaches [1, 6, 7] we do not investigate the structure of the CLB. Such an approach relies on our model of the CLB. Tests constructed according to our model of the CLB ensure the detection of all faults related to the inner structure of the CLB that can be detected by a single test pattern.

The testing of CLBs is similar to the classical TPG problem for ASIC circuits. But in the context of FPGA implementation, test vectors computed by classical TPG tool with the circuit netlist and adequate fault models produce a very low fault coverage [6]. This is mainly due to the fact that the circuit netlist used in the design phase before implementation into the FPGA does not contain any structural information on the final physical FPGA. A more accurate approach needs to consider – the real circuit mapped into a FPGA and a suitable fault model.

For a CLB, a fault may occur at the memory matrix, decoder, inputs and outputs of a CLB. A faulty matrix has some memory cells that are incapable of storing the correct logic values (stuck-at 1 or stuck-at 0 may occur at a memory cell). If a fault occurs at the decoder, then incorrect access, non-access or multiple access faults may occur. Consider an example. Let's say, the CLB under consideration accomplishes two inputs AND function. To detect all its single stuck-at faults, we have to construct 3 test patterns (0,1), (1,0), (1,1). The last possible test pattern (0,0) is not included into the test sequence. Let's say, this CLB because of a memory bit fault changes the function to NOR (Table 1). As we can see from the fifth column of Table 1, such a change of the function can be detected by the third test pattern. So, no additional test patterns are needed to detect such a change of the function. But if the CLB function because of the LUT memory bit fault changes to NOT(A) XOR B, such a fault cannot be detected by the first three test patterns. Only the test pattern (0, 0) which was initially unused can detect such a change of the CLB function.

Table 1. Test patterns for 2 inputs CLB

A	B	A AND B	A NOR B	NOT(A) XOR B
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1
0	0	0	1	1

Consider another example, presented in [7], where the CLB function L = (A AND B) OR C is given. All single stuck-at faults can be detected by four test patterns (Table 2): the first three test patterns {1, 2, 3} and one test pattern from the last three patterns {4a, 4b, 4c}. To detect the fault C Sa0, any of test patterns 4a, 4b or 4c can be used. There are unused four test patterns. Let's say, the CLB because of the LUT memory bit fault changes the function to LF = (A AND B) OR (NOT(B) AND C). An interesting situation arises. If an initial test set included the test pattern number 4a, this fault would be detected. But if an initial test set included a test pattern 4b or 4c, the fault would not be detected. There is no guarantee that the test pattern generator would include the test pattern number 4a. Therefore, in general, the fault is not detected by the initial four test patterns.

Table 2. Test patterns for 3 inputs CLB

N	A	B	C	NOT B	L	LF
1	0	1	0	0	0	0
2	1	0	0	1	0	0
3	1	1	0	0	1	1
4a	0	1	1	0	1	0
4b	1	0	1	1	1	1
4c	0	0	1	1	1	1

Note that a standard CLB usually has 4 address inputs. It means that more complicated functions than in case of 2 or 3 inputs can be constructed. Consequently, more changes because of memory bit faults can be made to the original function. These faults as we saw from the previous two examples cannot be detected by the test patterns devoted to detect all single stuck-at faults of the original function. Therefore we suggest to apply the exhaustive testing of the CLB function. Only the exhaustive testing of CLB ensures detecting of all possible function changes. If a CLB has 4 address inputs, so it has to be checked with all 2^4 different test patterns.

Now a single problem arises – how to generate exhaustive test patterns for each CLB of the FPGA mapped circuit. One possible solution is to use the classical stuck-at fault test pattern generator. Since stuck-at fault test generation tools are mature and highly efficient, it is conceivable that utilizing a stuck-at fault test generation tool for the exhaustive testing of CLB would be very effective. But the classical test pattern generator is able to generate test patterns only for single stuck-at faults. Therefore, there is a need for the circuit transformation. A circuit has to be transformed in such a way that this transformation would not change the function of the circuit and would compel the test pattern generator to test exhaustively every CLB in the circuit. A circuit transformation should transform a given circuit into different one before the test pattern generation. Then tests are generated for the transformed circuit. After that, the generated tests for the transformed circuit are directly applied without transformation for the original circuit.

We suggest to change every 2 inputs CLB by the circuit presented in Fig.1. This transformation satisfies all restrictions (the transformation of the circuit does not change the function of the circuit, the test pattern generator is compelled to test exhaustively every CLB in the circuit) presented in the above paragraph. Let's consider this circuit. As we see from Fig.1 every 2 inputs CLB is changed by the CLB itself and a multiplexer. Inputs of the CLB become controlling inputs of the multiplexer. The output of the CLB is connected to every data input of the multiplexer. Stuck-at faults are injected only on the data inputs of the multiplexer. So we have 8 stuck-at faults on

the inputs of the multiplexer. To test stuck-at faults on the first input of the multiplexer, the test generator has to assign to the controlling inputs test pattern (0, 0). But these inputs are also inputs of the CLB. So the inputs of the CLB also get the combination (0,0). To test stuck-at faults on the second input of the multiplexer, the test generator has to assign to the controlling inputs test pattern (0, 1). In such a way, the test generator provides all 4 combinations to the inputs of the CLB. Such a test is an exhaustive test of the address inputs of the CLB. Therefore the transformation presented in Fig.1 ensures the exhaustive testing of the 2 inputs CLB. The same principle of the transformation is applicable to any number inputs of the CLB.

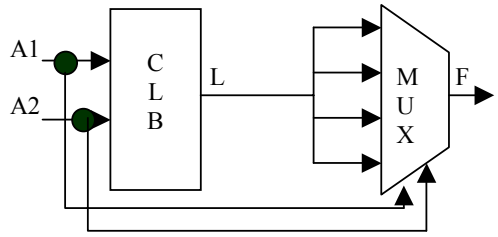

Fig. 1. The transformed 2 inputs CLB

3. Test pattern generation

It is necessary to notice that some faults in the proposed model are redundant. Consider the transformed CLB in Fig.1. As we remember the faults are injected on the data inputs of the multiplexer. Let's say CLB implements 2 inputs AND function. Then test pattern (0, 0) implies at the output of the CLB value 0 and this pattern for the multiplexer selects the first data input which has the value 0. Consequently, the fault Sa1 on the first data input of the multiplexer can be detected. But there is no test to detect the fault Sa0 on this input, because there is no possibility to provide the value 1. All other test patterns select other data inputs of the multiplexer. The test pattern (0, 1) implies the value 0 at the output of the CLB, selects the second data input of the multiplexer and can detect the fault Sa1 on this input. The fault Sa0 on the second data input can not be detected as well. The similar situation is with the other two test patterns. Initially we proposed to inject 8 stuck-at faults on the inputs of the multiplexer. But as we see, half of them is redundant. This is true for any function of the CLB. If 2 inputs CLB implements AND function redundant faults are:

- Sa0 on the first data input of a multiplexer;
- Sa0 on the second data input of a multiplexer;
- Sa0 on the third data input of a multiplexer;
- Sa1 on the fourth data input of a multiplexer.

If 2 inputs CLB implements OR function redundant faults are:

- Sa0 on the first data input of a multiplexer;
- Sa1 on the second data input of a multiplexer;
- Sa1 on the third data input of a multiplexer;
- Sa1 on the fourth data input of a multiplexer.

Thus redundant faults depend on the function of the CLB. Therefore there are two approaches for the test pattern generation: a) blind approach; b) targeted approach. In the blind approach, the functions of the CLB are not taken into account. Two faults are injected on every data input of the multiplexer. When the test pattern generator is not able to find a test for the target fault such a fault is declared as untestable. Of course, there is a possibility to make an improvement – if a test is already constructed for one fault of the input, another fault of this input can be declared as redundant and there is no need to generate a test for this fault. But we know that a test pattern generator is adopted to find test for the target fault. A test pattern generator has to spend a lot of time to conclude that the target fault is untestable. Therefore a blind approach has two deficiencies:

- spends a lot of time (sometimes hours) trying to construct a test for a redundant fault when before the test generation there is a possibility to identify that there is no test for such a fault.
- can not distinguish between redundant and untestable faults.

From the first sight the second conclusion may seem wrong. It is possible to argue that when a test constructed for one fault, the other fault of this input can be declared as redundant despite of the order of the generation for these faults. This is true when a test is constructed for one fault. But there can be situations when there is no possibility to find the test for both faults of the input because the input is not controllable or the output is not observable. In this situation, there is no possibility to distinguish which fault is redundant without considering a function of the CLB.

We believe the targeted approach is more preferred. In the targeted approach, the preprocessing step is taken before the test pattern generation. The functions of the CLB are analyzed and redundant faults are excluded from the list. The redundant faults will not be shown in the fault coverage statistics. In this case, if a test pattern generator is not able to construct a test for the target fault, this fault is really untestable.

A very similar approach for the exhaustive testing of high-level modules is applied in papers [8] where the model of the input pattern (IP) fault is presented. But these papers differ in the following:

- they do not target FPGA;
- the application of the multiplexer has the same goal but the multiplexer is adopted in a different manner.

We believe that our way of the adoption of the multiplexer is much simpler. A definition [8] of the IP fault model when a high-level module has a single output matches the faults used in our approach, therefore in the following we will use this term in order to distinguish from the classical stuck-at faults. Each IP fault of the module with a single output corresponds to the single input stimuli of the module. The detecting of all IP faults of the module corresponds to the exhaustive testing of the module.

4. Experiments

In the experiments, we used circuits from the ISCAS'85 benchmark suite. The circuits were mapped into FPGA using the Synopsys™ synthesis tool and Virtex™-II library. Tests for the transformed circuit were generated by the Synopsys™ test generation tool TetraMax™.

The test generation was carried out for three circuits from the benchmark suite ISCAS'85: c432, c880 and c5315. The numbers of CLBs of all three circuits are presented in Table 3.

Table 3. The CLBs of FPGA configured circuits

Circuit	2 inputs CLB	3 inputs CLB	4 inputs CLB	Total
C432	11	10	50	71
C880	21	21	79	121
C5315	109	62	319	490

The results of the test generation for IP faults are presented in Table 4. The total number of IP faults is the total number of the input stimuli of CLB implied by the exhaustive testing and can be counted as follows: the number of 2-inputs CLBs multiplied by 2^2 plus the number of 3-inputs CLBs multiplied by 2^3 plus the number of 4-inputs CLBs multiplied by 2^4. The total number of IP faults is calculated according to the figures in Table 3.

Table 4. Test generation for FPGA configured circuits

Circuit	Total	Detect	Rdun	Rdun % FPGA	Rdun % gate lev
C432	924	745	179	19.37	9.6
C880	1516	1442	74	4.87	6.1
C5315	6036	5392	644	10.67	15.8

The column under name "Rdun % (FPGA)" of Table 4 shows that every circuit has some fraction of redundancy. The proof of the existence of the redundancy costs some efforts. We would like to pay attention to the fact that tests were generated by the automatic test pattern generator. Such a generator usually abandons some faults due to run time restrictions. The similar situation was in our case. When a fault is abandoned, it is not clear if this fault is untestable or hard-to-detect. To resolve this situation the model of the transformed circuit was minimized by deleting multiplexers for CLBs that did not have undetected faults. Then the test generation for undetected faults was repeated. The process of minimization and generation was repeated several times. When the minimization did not lead to the solution the time of generation for a single fault was increased. The increase of time for a single fault was repeated until there were no abandoned faults. Of course, in some cases the time of the test generation increased from few seconds to hours, but therefore we can say for sure that undetected IP faults are redundant.

The last column of Table 4 is presented for comparison purposes. It shows a redundancy at gate level of the circuit. The numbers of this column were taken from the paper [8]. As we can see the tendency of the redundancy of IP faults for FPGA mapped circuits and their gate level equivalents is quite different. The only circuit c880 has similar numbers while the other two circuits have very different numbers. Such a result only confirms the fact that the FPGA implementation of the circuit is very different from its gate level equivalent.

The faults that are untestable in the given application of the FPGA mapped circuit may be testable in the other configuration. A configuration that will have no untestable faults is a tree like structure of the circuit. Therefore the untestable faults of the given application can be easily tested if the CLBs of the given application were reconfigured into a tree like structure. Of course, it needs to note that such a reconfiguration changes the initial function of the given application and a configuration process is excessively time-consuming.

Table 5. The comparison of test lengths

Circuits	No RC	No R1	No RG	No B2	No B3
C432	232	57	73	122	1123
C880	216	62	92	379	4955
C5315	332	130	168	1113	4598

R1 – The non-redundant ISCAS'85 benchmark circuit
RG – The gate level replacement of CLBs of FPGA
RC – FPGA configured circuit
B2 – Black Box model 2D matrix
B3 – Black Box model 3D matrix

The length and the coverage of the generated test sequence for the FPGA mapped circuit were compared with appropriate sizes of test sequences for other implementations of the circuit. The comparison of lengths of test sequences is presented in Table 5. We used for comparison the gate level tests of the original gate level circuit (R1), the tests for gate level replacement of the FPGA mapped circuit (RG). The latter implementation was included in the hope that it would have results of the test generation similar to the FPGA mapped circuit constructed of CLBs. As we will see later this hope did not stand – this implementation behave like the other gate level implementations. If were compared the lengths of tests only of these three implementations, the longest tests are of FPGA mapped circuits which were checked by the exhaustive testing of all input stimuli (RC). We also included for comparison tests generated according to the black box model of the circuit on the base of a two dimensional matrix (B2) [9] and on the base of a three dimensional matrix (B3) [10]. These tests are longer than for the FPGA mapped circuit, especially tests generated on the base of a three dimensional matrix. But we would like to pay attention that tests generated according to the black box model do not take into account the structure of the circuit. When the structure of the circuit is known it is possible to select quite a smaller subset of the initial set by means of the fault simulation.

The experiment shows that the exhaustive testing of CLBs cannot be established. Therefore the decision was made to analyze the reasons of this phenomenon. The circuit C432 was chosen for the experiment. The CLBs of the mapped into FPGA circuit C432 were leveled according to their distance to the primary inputs. Then the CLBs were grouped into groups according to their level and their number of inputs. The groups were formed of three consecutive levels of CLBs that an information would be presented in manageable quantities. The results are presented for each such group separately in Table 6. As we see only the first group of CLBs which is located nearest to the primary inputs has the 100% coverage of IP faults. Therefore the investigation was carried out to understand why IP faults of other groups are not fully detectable. All outputs of all CLBs were made like primary outputs. That ensures the propagation of all input IP faults

to the primary outputs. Such an experiment allows to distinguish between the problems of controlling the values on the inputs of the CLBs and propagating these values to the primary outputs. The experiment showed that none new group got a 100% coverage of IP faults. An increase was observed only for the groups of levels 10-12 and levels 13-15. The increase for the whole circuit was observed only from 80.63% to 84.30% of the IP fault coverage. 15.70% of IP faults were untestable because there was no capability to set an appropriate stimuli on the inputs of the CLB, 3.73% (84.30% - 80.63%) of IP faults were untestable because their effects were not propagated to the outputs of the circuit. It means that the function of the circuit restricts the capabilities to construct a certain stimuli on the inputs of the CLBs. The propagation of the stimuli to the primary outputs is a smaller problem.

Table 6. Results of C432

Levels		2 in	3 in	4 in	Total	(%)
1-3	Number of CLB	8	0	3	11	
	Total IP faults	32	0	48	80	
	Detected	32	0	48	80	100
	Undetected	0	0	0	0	
4-6	Number of CLB	0	4	12	16	
	Total IP faults	0	32	192	224	
	Detected	0	17	166	183	81.69
	Undetected	0	15	26	41	18.31
7-9	Number of CLB	0	5	14	19	
	Total IP faults	0	40	224	264	
	Detected	0	15	170	185	70.07
	Undetected	0	25	54	79	29.93
10-12	Number of CLB	0	1	13	14	
	Total IP faults	0	8	208	216	
	Detected	0	8	158	166	76.85
	Undetected	0	0	50	50	23.15
13-15	Number of CLB	3	0	8	11	
	Total IP faults	12	0	128	140	
	Detected	12	0	119	131	93.57
	Undetected	0	0	9	9	6.43
Total	Number of CLB	11	10	50	71	
	Total IP faults	44	80	800	924	
	Detected	44	40	661	745	80.63
	Undetected	0	40	139	179	19.37

When the tests were constructed for all considered implementations we crossed over the tests and the implementations. The results are presented in Table 7 and Table 8. Table 7 shows the results of the application of tests of various implementations to detect IP faults. We note that only testable IP faults were used in the experiment. These results were separated because the considered faults in this implementation are different from all the other implementations. The faults of the implementation RC denote IP faults of CLBs meanwhile the faults of all the other implementations are classical stuck-at faults. The tests generated according the black box model can detect 96.56% of IP faults (2D matrix) and 99.35% of IP faults (3D matrix). These numbers are quite higher than the numbers of the other implementations which were gate level except the tests that were targeted to the IP faults.

Table 8 includes the results of detecting stuck-at faults of various implementations by test patterns generated according to various models. The implementation RG (the gate level replacement of CLBs of FPGA) unlike the other implementations had redundant faults. These faults are excluded from the list of faults and their number is shown separately after the sign "+" in the line "#faults". We also included in this investigation two extra implementations: R2 – the synthesized original circuit on the base of library class.db, and R3 – the synthesized original circuit on the base of the library and_or.db. The tests were not constructed for these implementations separately.

Table 7. Testing of FPGA mapped circuits

Circuit	#IP faults	RC %	R1 %	RG %	B2 %	B3 %
C432	745	100	79.33	71.01	78.12	96.37
C880	1442	100	76.63	86.34	95.90	98.75
C5315	5392	100	92.10	94.20	99.29	99.92

Table 8. The test generation for stuck-at faults

Circuits		Implementations			
	Test	R1	R2	R3	RG
C432	# faults	507	420	460	430+9
	R1	100%	99.05%	99.78%	98.37%
	RG	99.21%	99.29%	100%	100%
	RC	99.80%	99.76%	100%	100%
	B2	96.05%	97.85%	98.69%	96.27%
	B3	100%	100%	100%	100%
C880	# faults	942	854	928	970+48
	R1	100%	99.88%	100%	98.25%
	RG	99.79%	100%	100%	100%
	RC	99.79%	99.88%	99.89%	100%
	B2	99.89%	99.88%	99.89%	99.89%
	B3	100%	100%	100%	100%
C5315	# faults	5248	3875	4130	3931+92
	R1	100%	99.74%	99.76%	99.47%
	RG	99.09%	99.41%	99.64%	100%
	RC	99.68%	99.95%	100%	100%
	B2	100%	100%	100%	100%
	B3	100%	100%	100%	100%

The tests B3 suit best of all for different implementations. But we do not have to forget that the purpose of these tests is just the same – to suit to any implementation of the given function. The diversity of tests B2 and tests RC is comparable. The tests B2 quite well suit for the implementation RC of the circuit, except the circuit C432. The tests RC (IP faults) show a very high fault coverage (aproximately 100%) for all circuits at the gate level implementation. It means that tests generated for IP faults suits very well for any implementation of the given circuit.

5. Conclusions

The paper proposes a testing approach for an FPGA configured circuit to implement a given application. This approach is based on the exhaustive testing of every logic cell. The exhaustive testing of logic cells ensures the detection of all defects in the inner structure of the logic cell. This does not depend on what configuration of the inner structure of the logic cell is used. The exhaustive testing of logic cells detects all defects that could be

detected by a single test pattern. But the length of the test is increased and a construction of such test requires some extra time to prove that certain combinations are untestable. The structure of the circuit restricts the capabilities of checking every logic cell in the circuit exhaustively. The experimental results showed that the exhaustive tests of the FPGA mapped circuit are 3.13 times longer in average than the tests for the equivalent gate level circuit. The tests of the gate level circuit can detect 87.90% of input patterns faults in average. The fault simulation experiments indicated that tests generated to detect input patterns faults have a strong tendency to detect non-targeted faults. These tests can detect 99.81% stuck-at faults in average of various implementations of equivalent gate level circuits. Our work proved that there is no need to consider internal faults of FPGA logic cells.

References

1. **Renovell M., Portal J.M., Faure P., Figueras J., Zorian Y.** Analyzing the Test Generation Problem for an Application-Oriented Test of FPGAs // Proceedings of the IEEE European Test Workshop. – Cascais, Portugal, 2000. – P. 157–162.
2. **Šeinauskas R., Bareiša E.** A Simulation-Based Test Patern Selection // Information technology and control. ISSN 1392-124X. – Kaunas: Technologija, 1997. – No.1(4). – P.13-18.
3. **Renovell M., Portal J.M., Faure P., Figueras J., Zorian Y.** Minimizing the number of Test Configurations for different FPGA families // IEEE 8th Asian Test Symposium. – Shangai, China, November 1999. - P.363-368.
4. **Michinishi H., Yokohira T., Okamoto T., Inoue T., Fujiwara H.** Testing for the Programming Circuits of LUT-based FPGAs // IEEE 6th Asian Test Symposium. – November 1997. – P. 242-247.
5. **Stroud C., Wijesuraya S, Hamilton C., Abramovici M.** Built-In Self Test of FPGA Interconnect // International Test Conference. Washington, USA, Oct. 18-23, 1998. - P.404-411.
6. **Renovell M., Portal J.M., Faure P., Figueras J., Zorian Y.** Some Experiments in Test Pattern Generation for FPGA-Implemented Combinational Circuits // Proceedings of IEEE 12th Brazilian Symposium on Integrated Circuit Design. – Manaus, Brazil, 2000. – P.3–8.
7. **Rebaudengo M., Sonza Reorda M., Violante M.** A New Functional Fault Model for FPGA Application-Oriented Testing, DFT 2002 // IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems. - Vancouver, Canada, November 6-8, 2002. – P.372–380.
8. **Blanton R. D., Hayes J. P.** On the properties of the input pattern fault model // ACM Transactions on Design Automation of Electronic Systems (TODAES)**.** Volume 8, Issue 1, January 2003. - P.108-124.
9. **Jusas V., Seinauskas R.** Automatic Test Patterns Generation for Simulation-based Validation. Proc. of the 8-th Biennal Baltic Electronics Conference. ISBN 9985-59-292-1. Tallinn Technical University. - Tallinn, Estonia, October 6-9, 2002. - P.295-299.
10. **Jusas V., Seinauskas R., Paulikas K.** Procedures for Selection of Validation Vectors on the Algorithm Level // Digest of papers of 2nd IEEE Latin-American Test Workshop. - Cancun, Mexico, February 11-14, 2001. - P.90-95.

Pateikta spaudai 2004 06 02

E. Bareiša, V. Jusas, K. Motiejūnas, R. Šeinauskas. FPGA loginių ląstelių testavimas // Elektronika ir elektrotechnika. - Kaunas: Technologija, 2004. - Nr. 7(56). – P.37–42.

Pagamintoms FPGA testuoti naudojamos kelios konfigūracijos ir konfigūruojami loginiai blokai (KLB) tikrinami parenkant visus rinkinius. Kai FPGA jau turi nustatytą funkciją, jai tikrinti visiško perrinkimo rinkiniai taip pat būtų labai pageidautini, nes reikia patikrinti visus išrinkimo lentelių bitus. Visiško perrinkimo testams sudaryti schema transformuojama. Kiekviena loginė ląstelė yra papildoma skirstikliu. Konstantiniai gedimai įvedami tik skirstiklio duomenų įėjimuose. Toks būdas leidžia naudoti klasikinį ventilio lygmens testų generatorių ir užtikrina visišką kiekvienos ląstelės perrinkimą. Pasiūlytas metodas buvo panaudotas ISCAS85 schemoms. Atlikti eksperimentai parodė, kad visiško perrinkimo rinkiniai gerai tinka įvairioms tos pačios schemos realizacijoms. Il. 1, bibl. 10 (anglų kalba; santraukos lietuvių, anglų ir rusų k.).

E. Bareiša, V. Jusas, K. Motiejūnas, R. Šeinauskas. Testing of FPGA Logic Cells // Electronics and Electrical Engineering. - Kaunas: Technologija, 2004. – No. 7(56). – P.37-42.

The manufacturing test procedure of RAM-based FPGAs uses several configurations and the exhaustive testing of all configurable logic blocks (CLB). The transformation of the circuit is applied during a test pattern generation. A multiplexer is added to every logic cell in such a way that it does not change a function of the circuit. The stuck-at faults are injected only on the data inputs of the multiplexer. Such an approach allows to use a classical gate level test pattern generator and ensures an exhaustive testing of every logic cell. The proposed approach was used to generate test sets for ISCAS85 benchmarks that were mapped into FPGA. We also conducted fault simulation experiments that show exhaustive test patterns are effective in detecting faults of different implementations of the same circuit. Ill.1, bibl. 10 (English, Abstracts in Lithuanian, English and Russian).

Э. Барейша, В. Юсас, К. Мотеюнас, Р. Шейнаускас. Тестирование логических блоков ПЛМ // Электроника и электротехника. – Каунас: Технология, 2004. –№7(56). - С.37-42.

Тестирование ПЛМ можно разделять на две категории: тестирование после производства и тестирование после загрузки конкретной функции. Тестирование после производства включает несколько конфигураций и подачу полного перебора тестовых наборов. В статье предлагается использовать эту же методику и для тестирования конкретной функции ПЛМ. Для того чтобы можно было использовать обычный генератор тестовых наборов для вентильного уровня каждый логический блок дополняется мультиплексором. Эксперименты были проведены для цифровых схем ISCAS85. Ил. 1, библ. 12 (на английском языке; рефераты на литовском, английском и русском яз.).