SAT-ATPG for Application-Oriented FPGA Testing

Robert Hülle, Petr Fišer, Jan Schmidt, and Jaroslav Borecký
Faculty of Information Technology
Czech Technical University in Prague
Prague, Czech Republic
Email: {hullerob, fiserp, schmidt, borecjar}@fit.cvut.cz

Abstract—**In this paper we propose a SAT-based ATPG algorithm for application-oriented FPGA testing. For this purpose, a novel fault model is introduced which combines the stuck-at fault model for interconnects testing with the bit-flip model for LUT testing. The concept of SAT-based ATPG enables integrating these two models easily. Fault coverage and fault dominance of the two models is discussed in this paper, yielding suggestions for using the proposed combined model.**

I. Introduction

Testing of digital circuits has become increasingly important. This is primarily due to higher ratios of faults occurring in contemporary deep-submicron designs, transient and intermittent faults caused by radiation [1], [2] and increasing complexity of designs.

To test a circuit off-line, a test sequence is typically computed by an Automatic Test Pattern Generation (ATPG) tool. We are referring to structural, circuit-specific tests. Many ATPG algorithms for gate-level structures were proposed since the 1980's [3], [4], [5]. However, as the complexity of designed circuits increases immensely, new ATPG algorithms are being developed [6], [7], [8] and hence the research in this field remains active.

Two types of ATPG algorithms prevail today: structural and SAT-based ones. *Structural* ATPGs are based on principles of the original D-algorithm [9]. Many sophisticated features have been developed, making them very efficient and scalable [4], [5], [8]. However structural ATPGs suffer from two major problems: their strong dependency on the fault model used, and difficulty to discover test vectors for hard-to-test faults or proving fault redundancy.

The second class of ATPGs is based on SAT (Boolean satisfiability) problem solving – the *SAT-based ATPGs*. They are more flexible in general, and are also very proficient in discovering redundant faults [10], [6], [7].

Designs based on Field-Programmable Gate Arrays (FPGAs), rather than custom logic, have become popular due to a relatively low price of FPGA chips and fast and cheap design development. At the present time, FPGAs are used in numerous end-product applications, not only for prototyping. Therefore, testing of FPGAs is increasingly important.

Testing of FPGAs is a problem much different from custom (ASIC) circuits testing. Typically, whole FPGA chips are tested using dedicated tests targeting their regular structure [11], [12], [13]. Particularly, the FPGA interconnect and look-up tables (LUTs) are tested separately, without considering the target application (the circuit implemented in the FPGA). In this way, the whole FPGA device is always tested, independently of its application. This approach is advantageous for *Manufacture-Oriented testing* of FPGA chips. However, an already programmed FPGA cannot be tested in this way, as this testing requires *reconfiguration* (testing configurations must be uploaded in place of the application) [14].

In *Application-Oriented testing* [14], [15], [16], [17], the circuit implemented in FPGA is tested instead of the FPGA fabric. In other words, the functionality of the logic programmed in FPGA is tested, not the device itself, and also the unused parts of the FPGA chip are not tested. The FPGA configuration is not modified for testing purposes here. For more details on Application-Oriented testing, see [18].

Testing of FPGAs, however, requires a specific fault model; it has been shown that the commonly used stuck-at fault model is insufficient [15], [19]. Particularly, LUT contents, interconnects, and device family specific features must be tested specifically. Note that in many previously published application-oriented FPGA testing approaches, standard stuck-at or gate-level fault models are used [17], [18], [20].

In this paper we propose a simplified, but rather universal fault model that can be used for application-oriented FPGA testing. Single faults (be it stuck-at or bit-flips) are assumed throughout the paper for simplicity, however, the model can be readily extended to support multiple faults. For this fault model, a SAT-based ATPG is presented and its properties are discussed.

II. Application-Oriented FPGA Testing

Since the FPGA fabric contains many different device-specific features, fault models used for custom design (ASIC) testing, such as the stuck-at fault model, are not suitable [15], [19]. Most past and contemporary FPGAs consist of:

1) look-up Tables (LUTs) of different sizes, typically contained in Configurable Logic Blocks (CLBs), together with flip-flops,
2) device specific primitives, such as fast carry chains (dedicated XOR gates) and multiplexers,
3) interconnects,
4) I/O and other communication blocks, and
5) special complex features, such as block-RAMs, DSP blocks, CPUs, etc.

In this paper we will focus on the first three types only, as their description can be generalised readily. Moreover, the last two types typically require special approaches to their testing [21], [11], [12], [22], [13], [23]. Also, only combinational circuits will be considered for simplicity; testing of flip-flops or sequential circuits in general would require a sequential ATPG process or a special design-for-testability (DFT) approach [24].

A. The Overall Algorithm (ATPG)

The SAT-based ATPG works by duplicating a part of a circuit and modelling the fault in the duplicated part [10]. For the typical stuck-at fault model, the faulty signal is duplicated and forced to have the stuck-at value. The rest of the circuit that is dependent on this signal (output cone) is also duplicated. The output of this duplicated circuit is then XORed with the original circuit and the Circuit Satisfiability Problem (CSAT) is then solved by reduction to a SAT instance, which is then solved by a SAT solver [25], [10]. Any satisfying variable assignment then represents a test vector. In the case where no such variable assignment exists, the fault is identified as redundant [10].

In addition to stuck-at faults, we are also considering single bit-flips in LUTs [2], [19]. We model these faults by duplicating a given LUT and injecting a fault in it, by flipping one bit in its memory. Then we duplicate the output cone of the affected LUT in the same way as with the stuck-at faults.

Not all faults need to be processed by a SAT solver, for we can use the fault dropping technique. The idea behind this technique is simple; one test vector usually covers more than one fault, so we need not compute test vectors for faults covered by this test vector. This not only decreases the testing time, but also speeds up test generation, because simple logic simulation is faster than solving, and especially generating the SAT instance.

B. The Proposed Combined Fault Model

As it was denoted in the Introduction, we aim at application-oriented testing. Thus, only the implemented circuit is tested, disregarding the unused parts of the chip. This scenario allows testing the interconnect using a standard stuck-at model, in contrast to transistor-oriented FPGA interconnection testing, where switching matrices, that are known only after the place&route phase, must be considered [21], [11]. Simple gates can also be tested using the stuck-at model. However, the stuck-at model is not sufficient for LUT testing [15], [19]. Therefore, a *single bit-flip* fault model is used for this purpose. This basically complies with the idea of [15], where stuck-at faults in all LUT cells were considered. However, one half of these faults were found redundant and had to be explicitly removed from the fault list. The bit-flip fault model directly eliminates this problem.

As shown in [14], [19], the initial circuit description (non-mapped netlist) is not suitable for ATPG purposes. However, the logic of the mapped circuit can be easily obtained from commercial FPGA synthesis tools [20]. As a result, the *mapped* netlist can be described as a multi-level Boolean network, where nodes are described in a Sum of Products

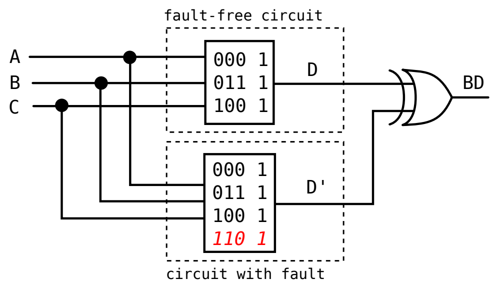

Fig. 1. Example of conceptual model of a bit-flip fault. The output of the circuit must be 1 to detect the fault. Bit-flip is distinguished by italics

(SOP) form. This network can then be directly used for test generation.

In this paper we describe the mapped logic by a network of *general nodes*. By general nodes we understand arbitrary single-output functions; however the approach can be easily extended for multi-output functions too (for the case of contemporary 2-output LUTs, for example).

For the implementation purposes, the BLIF format [26], where each node is described as a sum-of-products (SOP), is well suitable. Essentially, two types of nodes can be present in the mapped netlist:

1) a k-input LUT node, whose function (LUT content) can be described by a sum of minterms (SOM),
2) any other simple function (XOR, MUX, etc.) that can be described in a SOP form as well.

Summarised, the proposed fault model derived from the multi-level network of SOP nodes consists of:

1) all single bit-flips in the LUT, i.e., 2^k faults for a k-input node,
2) single stuck-at-0 and stuck-at-1 at all inputs and outputs of each node.

C. Generating Test for Different Fault Models

While the general ATPG algorithm remains the same, one step in particular differs across the fault models – the generation of SAT instances.

The difference is in how we model the fault itself. A stuck-at fault is modelled by disconnecting the faulty signal from the fault-free circuit and setting its value using a unit clause [10]. When modelling a bit-flip fault, there is no discontinuity in the faulty circuit. Instead, we model this fault by flipping the output for a particular LUT input vector.

For example, let us have a LUT described by 1-minterms $\{000, 011, 100\}$ and a bit-flip at address `110` which can be described by adding a minterm $\{110\}$. This situation can be seen in Figure 1.

D. Dominance between Stuck-at and Bit-Flip Faults

It can be shown that in a circuit with prevalent LUT nodes all testable stuck-at faults, that are located at LUT inputs or at the LUT output, are dominated by some bit-flip faults.

A stuck-at fault located at the output of a LUT may be not covered by a bit-flip in the LUT, if and only if there is no such

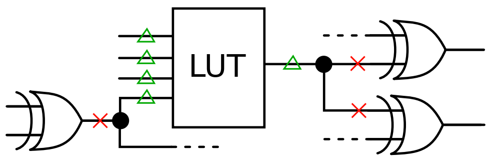

Fig. 2. Some stuck-at faults are always dominated by bit-flip faults (marked by triangle), while for other faults there is no such guarantee (marked by cross)

bit-flip, that would change the value of this signal in the same way as the stuck-at fault (for any input vector). However, such LUT would have a constant output, independent of its input, thus it would be redundant and so would be the stuck-at fault.

For stuck-at faults that are located at input signals of a LUT to be not covered by some bit-flip would mean that there exists no logical assignment of remaining LUT inputs, that would cause observable change at the LUT output for different values of the faulty signal. That would mean that the output of the circuit is independent of the signal value and thus the stuck-at would be redundant.

There is, however, no guarantee of dominance for stuck-at faults that are not adjacent to any LUT, be it at signals between two non-LUT elements or at signals after or before branching. Examples of such stuck-at faults are shown in Figure 2.

III. Experimental Results

For our experiments, we used 279 circuits from benchmarks MCNC, LGSynth'91 [27], LGSynth'93 [28], ISCAS'85 [29], ISCAS'89 [30] and IWLS 2005 [31]. For sequential circuits, combinational parts were extracted. The circuits were then synthesised by Xilinx Vivado 2015.2, for the Artix-7 architecture. After the synthesis step, we extracted the circuit structure from the EDIF format to BLIF [26].

We have measured the ratio of stuck-at faults that are covered by bit-flip faults, and the ratio of bit-flip faults that are covered by stuck-at faults. Results for several selected circuits can be seen in Table I; average values obtained from all the tested circuits are shown in the last row.

We have found, that almost all stuck-at faults are dominated by bit-flip faults. An example of a circuit, where there are stuck-at faults that are not dominated by a bit-flip fault, is circuit `barrel16a`, which contains primitives, such as multiplexers.

For bit-flip faults, we have observed a dominance by stuck-at faults ranging from 17% to 100%, with the average of 69.4% and the median of 72.6%.

We have found a surprisingly large set of redundant faults; the observed average ratio of redundant faults was 0.33% for stuck-at faults, 10.15% for bit-flip faults and 7.42% for all faults, as can be seen in Table I. Such a high amount of redundant bit-flip faults is due to a smaller controllability and observability of these faults, i.e., the probability that such test vector exists, so all inputs of a given LUT are set to excite the fault *and* it is propagated to a primary output. Note that propagation to an output is equivalent to stuck-at faults propagation, but excitation needs more signals to be set to a specific value.

We have also examined how do the final test lengths differ, for the two fault models and for different orderings of faults. Results of measurements for few selected circuits can be seen in Table II. We have found that the ordering of faults has a small impact on the number of testing vectors. When we look at the ratio of these two orderings, we see that it ranges from 0.81 up to 1.11, with the average of 0.97 and standard deviation of 0.04. This means that there is no significant difference between these two orderings; the difference is just due to the algorithmic noise [32].

IV. Conclusions

In this paper, we have demonstrated a SAT-based ATPG capable of directly working with the bit-flip fault model in addition to the usual stuck-at model.

We have examined the two fault models in the context of application-oriented FPGA testing and their interaction in respect to their mutual dominance. We have found that most of stuck-at faults are dominated by bit-flip faults. The cases where stuck-at faults are not dominated include FPGA primitives, such as multiplexers. Bit-flip faults, on the other hand, are generally dominated by stuck-at faults to much smaller degree, ranging from as low as 17%.

We conclude, that for a complete coverage of these two fault models, both must be considered. However, most of stuck-at faults are dominated by bit-flip faults. These are easily identifiable from the circuit structure, as they are adjacent to LUTs and their dominance is independent of the circuit function. They constitute majority of stuck-at faults, thus their omission may lead to a significant ATPG speed-up. To reach a complete fault coverage, structurally not dominated stuck-at faults must be considered too, since some of them are not covered by bit-flip faults (they are not dominated functionally).

We have also examined the overall lengths of test generated with the fault-dropping technique for two orderings of faults and have found no significant impact of fault ordering on the number of generated test vectors.

Acknowledgement

This work was partially supported by the grant GA16-05179S of the Czech Grant Agency, "Fault Tolerant and Attack-Resistant Architectures Based on Programmable Devices: Research of Interplay and Common Features" (2016-2018). Computational resources were provided by the MetaCentrum under the program LM2010005 and the CERIT-SC under the program Centre CERIT Scientific Cloud, part of the Operational Program Research and Development for Innovations, Reg. no. CZ.1.05/3.2.00/08.0144. This research has been in part supported by CTU grant SGS16/121/OHK3/1T/18.

References

[1] R. Velazco, P. Fouillat, and R. Reis, *Radiation Effects on Embedded Systems*. Springer Netherlands, 2010.

[2] M. Nicolaidis, *Soft Errors in Modern Electronic Systems*, ser. Frontiers in Electronic Testing. Springer US, 2010.

TABLE I
REDUNDANT FAULTS AND COVERAGE OF NON-REDUNDANT FAULTS IN SELECTED CIRCUITS

	stuck-at				bit-flip				total	
circuit	#	redundant [%]	testable	covered [%]	#	redundant [%]	testable	covered[%]	#	redundant [%]
alu4	2484	0.85	2463	100	9060	17.27	7495	68	11544	13.7
barrel16a	1246	0	1246	91.1	3208	1.5	3160	90.8	4454	1.08
cm42a	136	0	136	100	160	0	160	100	296	0
cmb	162	0.62	161	100	342	13.16	297	31	504	9.13
dalu	2492	6.86	2321	100	8056	35.74	5177	87.4	10548	28.9
des	6626	0.09	6620	100	15960	20.4	12704	95.6	22586	14.4
dsip	8960	0	8960	100	17944	0	17944	87.5	26904	0
mux	118	0	118	100	320	0	320	23.1	438	0
s9234	5098	0.53	5071	98.9	10584	17.01	8784	82.2	15682	0
average		0.33		99.6		10.15		69.5		7.42

TABLE II
LENGTH OF TEST FOR DIFFERENT FAULT ORDERING

circuit	stuck-at	bit-flip	SA,BF	BF,SA	$\frac{\text{SA,BF}}{\text{BF,SA}}$
alu4	413	2844	2866	2871	0.998
barrel16a	112	766	745	804	0.927
cm42a	16	16	16	16	1
cmb	24	227	218	228	0.956
dalu	158	1181	1165	1190	0.979
des	296	2232	1944	2214	0.878
dsip	313	4190	4052	4212	0.962
mux	33	306	297	306	0.971
s9234	375	2649	2618	2647	0.989
average					0.971

[3] P. Goel, "An implicit enumeration algorithm to generate tests for combinational logic circuits," *IEEE Transactions on Computers*, vol. C-30, no. 3, pp. 215–222, March 1981.

[4] H. Fujiwara and T. Shimono, "On the acceleration of test generation algorithms," *IEEE Transactions on Computers*, vol. C-32, no. 12, pp. 1137–1144, Dec 1983.

[5] M. Schulz, E. Trischler, and T. Sarfert, "SOCRATES: a highly efficient automatic test pattern generation system," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 7, no. 1, pp. 126–137, Jan 1988.

[6] S. Eggersglüß and R. Drechsler, "Robust algorithms for high quality test pattern generation using Boolean satisfiability," in *IEEE International Test Conference*, Nov. 2010, pp. 1 –10.

[7] H. Chen and J. Marques-Silva, "A two-variable model for SAT-based ATPG," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 12, pp. 1943–1956, Dec 2013.

[8] R. Ubar, L. Jurimagi, E. Orasson, G. Josifovska, and S. Oyeniran, "Double phase fault collapsing with linear complexity in digital circuits," in *Euromicro Conference on Digital System Design (DSD)*, Aug 2015, pp. 700–705.

[9] J. Roth, "Diagnosis of automata failures: A calculus and a method," *IBM Journal of Research and Development*, vol. 10, no. 4, pp. 278–291, July 1966.

[10] T. Larrabee, "Test pattern generation using Boolean satisfiability," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 11, no. 1, pp. 4–15, Jan. 1992.

[11] C. Stroud, S. Wijesuriya, C. Hamilton, and M. Abramovici, "Built-in self-test of FPGA interconnect," in *International Test Conference*, Oct 1998, pp. 404–411.

[12] M. Renovell, J. Figueras, and Y. Zorian, "Test of RAM-based FPGA: methodology and application to the interconnect," in *15th IEEE VLSI Test Symposium*, Apr 1997, pp. 230–237.

[13] M. Renovell, J. Portal, J. Figuras, and Y. Zorian, "Minimizing the number of test configurations for different FPGA families," in *8th Asian Test Symposium (ATS'99)*, 1999, pp. 363–368.

[14] M. Renovell, M. Portal, J., P. Faure, J. Figueras, and Y. Zorian, "Analyzing the test generation problem for an application-oriented test of FPGAs," in *IEEE European Test Workshop*, May 2000, pp. 75–80.

[15] M. Rebaudengo, S. Reorda, Matteo, and M. Violante, "A new functional fault model for FPGA application-oriented testing," in *17th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, 2002, pp. 372–380.

[16] M. Rozkovec, J. Jeníček, and O. Novák, "Application dependent FPGA testing method," in *13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools (DSD)*, Sept 2010, pp. 525–530.

[17] K.-H. Diener, G. Elst, E. Ivask, J. Raik, and R. Ubar, "FPGA design flow with automated test generation," in *11th Workshop on Test Technology and Reliability of Circuits and Systems*, 1999, pp. 120–123.

[18] M. Renovell, "Some aspects of the test generation problem for an application-oriented test of SRAM-based FPGAs," *Journal of Circuits, Systems and Computers*, vol. 12, no. 02, pp. 143–158, Feb. 2003.

[19] J. Borecký, M. Kohlík, P. Kubalík, and H. Kubátová, "Fault models usability study for on-line tested FPGA," in *14th Euromicro Conference on Digital System Design (DSD)*, Aug 2011, pp. 287–290.

[20] M. Renovell, M. Portal, J., P. Faure, J. Figueras, and Y. Zorian, "TOF: a tool for test pattern generation optimization of an FPGA application oriented test," in *9th Asian Test Symposium*, Dec. 2000, pp. 323–328.

[21] H. Michinishi, T. Yokohira, T. Okamoto, T. Inoue, and H. Fujiwara, "A test methodology for interconnect structures of LUT-based FPGAs," in *The Fifth Asian Test Symposium*, Nov 1996, pp. 68–74.

[22] M. Renovell, J. Portal, J. Figueras, and Y. Zorian, "SRAM-based FPGA's: testing the LUT/RAM modules," in *International Test Conference*, Oct 1998, pp. 1102–1111.

[23] W.-K. Huang, F. Meyer, N. Park, and F. Lombardi, "Testing memory modules in SRAM-based configurable FPGAs," in *International Workshop on Memory Technology, Design and Testing*, Aug 1997, pp. 79–86.

[24] M. Renovell, P. Faure, J. Portal, J. Figueras, and Y. Zorian, "IS-FPGA: a new symmetric FPGA architecture with implicit scan," in *International Test Conference*, 2001, pp. 924–931.

[25] N. Eén and N. Sörensson, "An extensible SAT-solver," in *Theory and Applications of Satisfiability Testing*, ser. Lecture Notes in Computer Science, E. Giunchiglia and A. Tacchella, Eds. Springer Berlin Heidelberg, 2004, vol. 2919, pp. 502–518.

[26] University of California, Brekeley, "Berkeley logic interchange format (BLIF)," 2005.

[27] S. Yang, "Logic synthesis and optimization benchmarks user guide: Version 3.0," Jan. 1991.

[28] K. McElvain, "IWLS'93 Benchmark Set: Version 4.0," Tech. Rep., May 1993.

[29] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran," in *Proceedings of IEEE Int'l Symposium Circuits and Systems (ISCAS'85)*. IEEE Press, Piscataway, N.J., 1985, pp. 677–692.

[30] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *IEEE International Symposium on Circuits and Systems, 1989*, May 1989, pp. 1929–1934 vol.3.

[31] C. Albrecht, "IWLS 2005 benchmarks," Tech. Rep., June 2005.

[32] W. Shum and H. Anderson, Jason, "Analyzing and predicting the impact of CAD algorithm noise on FPGA speed performance and power," in *International ACM Symposium on Field-Programmable Gate Arrays*, 02 2012, pp. 107–110.