ELECTRONICS AND ELECTRICAL ENGINEERING
ISSN 1392 – 1215 2008. No. 7(87)
ELEKTRONIKA IR ELEKTROTECHNIKA

T 171 MIKROELEKTRONIKA
MICROELECTRONICS

LSFR and BIST based Delay Test for ASIC and FPGA

V. Abraitis, Ž. Tamoševičius
The Department of Software Engineering, Kaunas University of Technology, Studentų str. 50, LT-51368 Kaunas, Lithuania, phone: +370 37 300361; e-mail: abravida@elen.ktu.lt

Introduction

The complexity of present-day electronic devices has risen to millions of gates, and the chips are therefore becoming untestable by standard manufacture external Automated Test Equipment (ATE) testers. The test lengths for Very Large Scale Integration (VLSI) ASIC are rapidly increasing, as are the testing times and the ATE memory requirements. Other hardly testable range of integrated circuits is Programmable Logic Devices (PLD). FPGA and Complex Programmable Logic Devices (CPLD) represent a class of them. User can program the final function for such device. The reconfigurability of such circuits is taking more significance for System-On Chip (SOC) designers. For such kind of reasons PLDs are very popular. At this time PLDs are used in civil industry, medicine, military area and space technologies. The testability and high reliability questions are very important for such reasons. But internal PLD architecture is very specific and already known methods for ASIC can't fully check them. So why we need new methods or we have to modify the old and adopt them for PLD testing. Problems related with PLD testing are discussed in proceeding articles [1 – 3].

Hence the built-in self-test (BIST) can resolve some testability problems. The circuit is able to test itself by BIST without using any ATE equipment, or when used together with an ATE, BIST significantly reduces the test time and tester memory demands. Moreover the user of PLD can complement the own function with BIST ant program all it in to same PLD.

Many BIST techniques have been developed [4, 5]. The vast majority of them use a pseudo-random pattern generator (PRPG) to produce test vectors that detect the easy-to-detect faults, which mostly represent more than 90% of the total faults. For the remaining faults, test vectors are either applied externally, or they are generated by the BIST structure itself.

In this paper we use linear feedback shift registers (LFSR) and cellular automata (CA), due to their simplicity and good properties concerning implementation space demands and the good transition fault coverage for circuits implemented in to ASIC and FPGA.

A general BIST structure is shown in Fig. 1. The patterns are generated by a test pattern generator (TPG), then they are fed to the circuit-under-test (CUT) and the circuit's responses are evaluated by test response evaluator (TRE) which gives test result or signature of the test result. All this structure is controlled by the BIST controller.

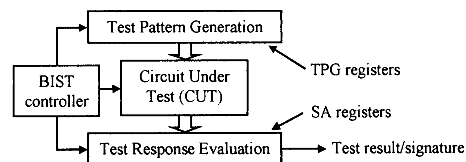

Fig. 1. BIST structure

The design of the TPG is of key importance for the whole BIST, since it determines the fault coverage achieved. A simple LFSR often cannot ensure satisfactory fault coverage, thus it has to be augmented in some way. The LFSR code word sequence is modified in some approaches to produce patterns that detect more faults. These methods imply reseeding the LFSR during the test, or possibly the generating polynomial is also modified, or the LFSR patterns are modified by an additional logic [5].

The proper choice of a PRPG is very important. It is desirable to detect as many faults as possible by the PRPG, so that the additional logic is maximally reduced. We introduce statistics on the transition fault coverage for the ITC'99 B benchmarks, using different PRPGs for transition faults in the FPGA environment.

Architecture of FPGA

Manufacturers are selling already checked chips, but user tests are important too, because a set of different reasons, like: devices are stored for a time, can be damaged by transportation or damaged by some harmful emission. So PLDs must be checked before using them in responsible applications. But test for PLD is reasonable only after they are programmed. For this we can't use traditional automatic test patterns generators (ATPG) directly. Such generators are used for traditional ASICs and test made by them can't fully check PLDs. It is because of different realizations and it was proved by experiments [2]. Such tests are not estimating internal PLD physical structure [6]. ATPGs are not estimating possible failures in the memory

cells of PLD, so why we need to modify original circuit and change it into model with the same functionality. And if we want to make a model of the real circuit with the same functionality, at first we have to analyze the internal structure and architecture of PLDs.

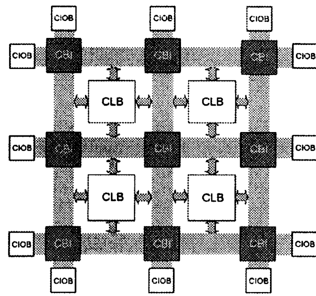

Fig. 2. Simplified architecture of PLD

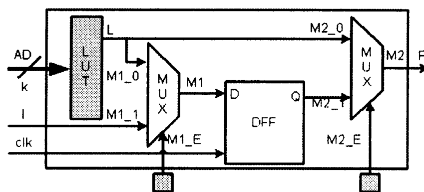

Fig. 3. Simplified architecture of CLB

Usually each PLD's family separates from others by some features, but almost all of them consist of a matrix of configurable logic blocks (CLB) and configurable blocks of inputs and outputs (CIOB). All CLB and CIOB are connected to each other by configurable blocks of interconnections (CBI) and a lot of conductors (Fig.2.). If we'll discus only about FPGA, then all configurations are realized by loading the SRAMs with logic zero or logic one. And almost all SRAM based PLDs have most same internal structure of CLB (Fig.3.). CLB has three main components: a look-up tables (LUT), multiplexers and D flip-flops. The difference is only that each manufacturer uses different sizes and quantities of LUTs, multiplexers and D flip-flops in one CLB. And sometime they use some simple combinational logic, like OR, AND, XOR and others.

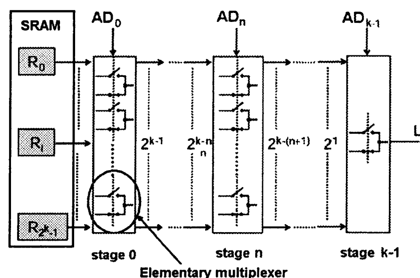

Fig. 4. Common structure of LUT

Grey boxes in Fig.3 represent configuration memory cells (SRAM in the FPGA). A LUT can be programmed to implement any k-input combinational function or to work as a 2k bit's of RAM. The function of LUTs depends on Truth table, with is saved in SRAM cells. The CLB internal interconnections are configurable by corresponding SRAM cells too. CBI consists of commutating transistors and each of them is controllable by appropriate SRAM cell. All FPGAs are programmable by writing appropriate value to due SRAM cell. Such set of values is named configuration.

Analyzing FPGA devices and possible their faults we have to analyze the internal structure of LUTs. The common structure is showed in Fig.4. R0 – R2k-1 are SRAM cells used to save the Truth table of the function of LUT. AD controls multiplexers and so at the same time only one Ri can be connected to the output L.

Almost all SRAM based FPGAs have most same architecture and internal CLB and LUT structure, like it is showed in figures 2,3 and 4.

The transition fault model of FPGA

It is usual to model the functionality of electronic devices, but it is possible to model faults too. To do it we need special models of real devices. Most popular are stuck-at, path delay (transition) and element delay fault models. The fault models describe what must be checked, what faults are possible in the concrete node.

Realistically, defects can be divided by derivation to processing defects, defects of silicon, time depending faults, packaging. Almost all these faults can be modeled by one wire fixation, open and shortly connected transistor, transition and delay models.

Making transition and delay faults models there can be mentioned two kinds of faults: STF – slow to fall; STR – slow to rise. STR, wherever it is in the circuit, can be took when transition from 0 to 1 (from 1 to 0, in transition to 0 case) does not effects any output or trigger of the circuit in particular set time period. There is only one difference between transition and delay faults: particular set time range for delay fault for transition. In case of transition fault it is took that time for transition is infinite.

There are couple of vectors (V1, V2) used for transition faults test. They can't be equal. Also all R meanings (Fig.4) can't be same; at least one of them must be equal to 0 and at least one to 1. These are the main requirements for testing LUT component [7]. V1 is a vector for initiation, and V2 is the transition vector, which not only initiate transition from one meaning to another, but also generates wave of transitions in all net to the output of circuit or to scan trigger, where this transition can be seen and tested. Based on this we can state that device will be fully tested if rising and falling fronts will be formed on all possible nets. Then rising front could check transition to logical 1 on the concrete path, and falling front could check transition to logical 0 faults on the specific path. Some faults can't be checked for circuit's function, because there originate some limitations and it is impossible to set one or other meaning in some node. So the transition can't be observable.

The hardest thing in programmable logical devices is to check LUTs. Because of these blocks inner structures and originate main differences between traditional ASIC circuits and PLDs. Generally the test quality can be represented by quantity of checked paths in whole circuit. In FPGA case it will be the total number of inputs on the firs stage of multiplexers (Fig.4). But to check multiplexer

it is necessary to use all possible combination on the inputs and it is hard to do. There will be impossible to save such amount of test vectors and reactions into them. To resolve this problem we will use BIST and analyze how many possible faults can be checked in this way.

The PRPG Structure in FPGA implementation

Generally, PRPGs are simple sequential circuits generating code words, according to the generating polynomial. These code words are then either fed directly to the CUT inputs, or they are modified by some circuitry.

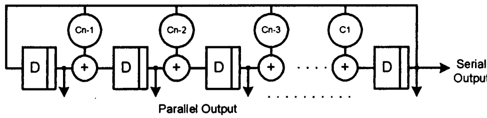

Fig. 5. LFSR implementation

The most common PRPG structures are linear feedback shift registers or cellular automata. An n-bit LFSR is a sequential circuit consisting of D flip-flops and XOR gates generating code words of a cyclic code. The structure of an n-bit LFSR-I (with internal XORs) is shown in Fig. 5. The register has n parallel outputs corresponding to the outputs of the D flip-flops, and one flip-flop output can be used as a serial output of a register.

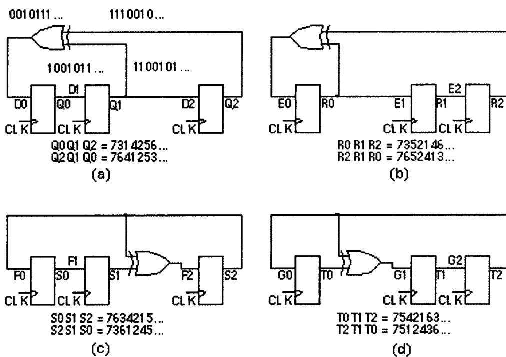

Fig. 6. Four linear feedback shift registers for every primitive polynomial

The coefficients c_1 till c_{n-1} express whether there exists (1) a connection from the feedback to the corresponding XOR gate or no connection (0).

The initial state of the register is called the seed. The sequence of code words produced by an LFSR can be described by a generating polynomial g(x) in GF.

The second LFSR type, the LFSR-II is implemented with XORs in the feedback. Its generating polynomial is dual to the LFSR-I polynomial, so only LFSR-I will be considered in this paper.

For every primitive polynomial there are four linear feedback shift registers (LFSRs) (Fig.6). There are two types of LFSR; one type uses external XOR gates (type 1) and the other type uses internal XOR gates (type 2). For each type the feedback taps can be constructed either from the polynomial g(x) or from its reciprocal, g*(x). The LFSRs in this figure correspond to $g(x) = 1 + x + x^3$ and $g^*(x) = 1 + x^2 + x^3$. The sequences shown in Fig.6 are for each register initialized to binary '111': (a) Type 1, g*(x), (b) Type 1, g(x), (c) Type 2, g*(x), (d) Type 2, g(x).

Cellular automata are sequential structures similar to LFSRs. Their periods are often shorter, but code words generated by CA are sometimes more suitable for test patterns with preferred numbers of ones or zeros at the outputs. An example of a CA performing multiplication of the polynomials corresponding to code words by the polynomial x+1 is shown in Fig. 7.

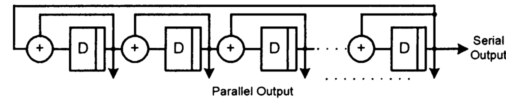

Fig. 7. CA implementation

In general, pseudo-random patterns generated by a CA have a more random nature than those generated by an LFSR. The weights of the particular PRPG outputs (i.e., the ratios of zeroes and ones) are balanced in LFSRs approaching the value 0.5. Cellular automata often have the weights misbalanced, according to the seed.

Experiments

We have performed experiments on the ITC'99 benchmarks, to determine fault coverage achieved by a pseudo-random test sequence generated by a two PRPG's – LFSR and CA. The experiments were made in this way: (1) test vector pairs for transition faults were generated by our made software. The software simulates the LFSR and CA and calculates test vector pairs; (2) fault coverage of the generated test vector pairs for benchmark circuits implemented in the FPGA we checked with industrial software "Tetra Max". Obtained results are presented in table 1.

Short description of the table 1: a 1st column is a name of the benchmark circuits. Second column gives a number of transition faults in the corresponding benchmark circuit. Third column represents number of test patterns for circuit synthesized for FPGA, for original circuit, generated by LFSR and CA respectively. Fourth column represents fault coverage for mentioned test pattern generators. Sixth column shows how much test vectors in the generated set of vectors from ATPG are equal to generated set from LFSR and CA (seventh column).

Table 1. Experimental results.

Cir-	Total	Number of test patters				Coverage [%]				LFSR rep. pat.		CA rep. pat.	
cuit	Faults	Mux	Orig.	LFSR	CA	Mux	Orig.	LFSR	CA	Mux	Orig.	Mux	Orig.
B01	260	59	35	13	13	63.46	48.84	20	20	11	3	11	3
B03	660	120	71	46	46	76.67	63.33	5.60	5.60	0	0	0	0
B04	2632	245	120	13372	13372	67.29	58.02	5.78	5.78	0	0	0	0
B06	308	57	40	6	6	65.58	55.19	13.31	13.31	0	0	0	0
B13	1644	237	109	43735	43735	74.51	61.55	16.46	16.46	8	2	8	2

Summary

When we look in table 1 prima facie is that our proposed methods are very slow when we compare to Mux and Plain tests. Do not forget please the environment in what we are using generated tests – a BIST for FPGA. This means we have a very limited space in the chip and we should pay a big attention to the physical size to store the test vectors in the chip. To store the test vectors, generated by ATPG, we need to synthesize a logical structure which will take a lot of space in the chip. When we use our proposed method to implement test pattern generator in to the chip, an area overhead in the chip is very low – a couple of filp-flops and a couple of XOR elements. In the other hand, we used a very simple test vector generators and we didn't use a reseeding, but we get a very promising results: (1) patterns generated by ATPG is not very useful to use in the BIST while we get a very high area overhead; (2) area overhead is very low when we use a LFSR or CA as in-circuit test patterns generators.

A low area overhead and good speed of the designed BIST strictly depend on the nature of the circuit. Pseudo-random testability of a particular circuit strictly depends on the number of hard-to-detect faults. It is possible to apply an unmodified sequence of LFSR code words to fully test some circuits in a reasonable number of cycles, while some other circuits are particularly untestable by this way.

It is not possible to compute a proper LFSR seed and/or generating polynomial analytically for practical examples, due to the complexity of this problem. Thus, in practice we repeatedly reseed the polynomial and conduct the fault simulation several times, while we pick out the best seed for further processing.

Our future work will involve reseeding. We will try to implement the reseeding to make fault coverage more attractive.

References

1. **Abraitis V., Bareiša E.** The Fault Model of Programmable Logic Block // Electronics and Electrical Engineering. – 2005. – No. 6(62). – P. 52–56.
2. **Abraitis V., Bareiša E., Benisevičiūtė R.** The Testing Methods of Programmable Integrated Circuits // Electronics and Electrical Engineering. – 2003. – No. 5(47). – P. 43–47.
3. **Bareiša E., Jusas V., Motiejūnas K., Šeinauskas R.** Testing of FPGA Logic Cells // Electronics and Electrical Engineering. – 2004. – No. 7(56). – P. 37–42.
4. **Bareiša E., Jusas V., Motiejūnas K., Šeinauskas R.** Functional Digital Systems Testing. ISBN 9955-25-008-9. – Kaunas: Technologija, 2006. – P. 281.
5. **Jha N. K., Gupta S.** Testing of Digital Systems // ISBN 0 521 77356 3. – Cambridge University Press, 2003. – P. 1016.
6. **Renovell M., Figueras J., Zorian Y.** Test of RAM-Based FPGA: Methodology and Application to the Interconnect // IEEE VLSI Test Symposium. – 1997. – P. 230–237.
7. **Girard P., H′eron O., Pravossoudovitch S., Renovell M.** Requirements for Delay Testing of Look-Up Tables in SRAM-Based FPGAs // Eighth IEEE European Test Workshop. – 2003. – P. 147–152.

Received 2008 02 02

V. Abraitis, Ž. Tamoševičius. LSFR and BIST based Delay Test for ASIC and FPGA // Electronics and Electrical Engineering. – Kaunas: Technologija, 2008. – No. 7(87). – P. 45–48.

Transition delay testing of sequential circuits in a clocked environment is analyzed. There are presented two test pattern generator methods for built in self testing of the circuit implemented as Application Specific Integrated Circuit (ASIC) and Field Programmable Gate Array (FPGA). Cellular automaton and Linear Feedback Shift Register (LFSR) structures are used for test sequence generation. The circuits are tested as the black boxes under Transition fault model. Experimental results of the test pattern generation methods are presented and analyzed. Results compared with exhaustive test of transition faults for ASICs and programmable integrated circuits with given configuration. Ill. 7, bibl. 7 (in English; summaries in English, Russian, Lithuanian).

В. Абрайтис, Щ. Тамошявичюс. Генерирование тестовых последовательностей для заказных и программируемых интегральных схем со встроенными схемами самотестирования, используя структуры линейного регистра сдвига с обратными связями // Электроника и электротехника. – Каунас: Технология, 2008. – № 7(87). – С. 45–48.

Рассматриваются модели неисправностей переключения последовательных интегральных схем (ИС). Представлены два метода генерирования тестовых последовательностей для заказных (ASIC) и программируемых интегральных схем со встроенными схемами самотестирования. Для генерирования тестовых последовательностей использованы структуры клеточного автомата и регистра сдвига с обратными связями. Схемы тестированы как чёрные ящики используя модели неисправностей переключения. Представлены результаты экспериментов с тестирования методов генерирования тестовых последовательностей. Результаты сравнены с тестами неисправностей переключения, тестируя заказные и программируемые интегральные схемы. Ил. 7, библ. 7 (на английском языке; рефераты на английском, русском и литовском яз.).

V. Abraitis, Ž. Tamoševičius. Testinių rinkinių sudarymas vėlinimo gedimams tikrinti save testuojančiose integrinėse schemose, realizuotose ASIC ir FPGA, naudojant postūmio registro su tiesiniu grįžtamuoju ryšiu struktūras // Elektronika ir elektrotechnika. – Kaunas: Technologija, 2008. – Nr. 7(87). – P. 45–48.

Analizuojamas nuosekliųjų schemų perjungimo gedimų testavimas. Pristatomi du testinių rinkinių sudarymo metodai save testuojančioms schemoms realizuotoms užsakomuosiuose integrinių schemų lustuose (ASIC) ir programuojamuosiuose lustuose. Testinėms sekoms generuoti naudojamos ląstelinio automato ir postūmio registro su grįžtamuoju ryšiu struktūros. Testinės schemos buvo testuojamos kaip juodos dėžės naudojant perjungimo gedimų modelį. Pristatyti ir išanalizuoti testinių rinkinių generavimo metodų eksperimentinių tyrimų rezultatai. Rezultatai palyginti su perjungimo gedimų testais, skirtais ASIC ir programuojamiesiems lustams, užprogramuotiems vykdyti reikiamą funkciją. Il. 7, bibl. 7 (anglų kalba; santraukos anglų rusų ir lietuvių k.).