ISSN 1392 – 1215 ELECTRONICS AND ELECTRICAL ENGINEERING 2008. No. 1(81)
ELEKTRONIKA IR ELEKTROTECHNIKA

T 120 SYSTEM ENGINEERING, COMPUTER TECHNOLOGY
SISTEMŲ INŽINERIJA, KOMPIUTERINĖS TECHNOLOGIJOS

The Transition Fault Model of Programmable Logic

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

Recently the need for specialized devices is going greater and greater. But need for such devices in the market are in small quantities. To manufacture something in small quantities mostly in all cases is unprofitable. This problem can be resolved using universal devices. Such devices are made so, what they don't have a special function yet. User can chose and program the final function for device. Such devices can be manufactured in big quantities and if we will analyze economical aspect it increases the profit. The final product will be device with specialized function required by user. FPGA (Field Programmable Gate Array) and CPLD (Complex Programmable Logic Device) represent a class of PLD (Programmable Logic Device). PLDs are such universal devices. Programmable area and speed are growing very fast with years, when price decreases at the same time. The reconfigurability of such circuits is taking more significance for System-On Chip (SOC) designers. Actually, reconfigurable logic gives to SOC system designers much greater flexibility. For such kind of reasons PLDs are very popular. User can choose PLD from big variety, depending from size, speed, amount of logic, triggers and memory or power supply.

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. 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 [1,2,3].

One of the most important PLD types is the SRAM-based FPGA. In such FPGA, an array of logic cells and interconnections can be configured to implement the desired function by loading the SRAMs.

Aims of FPGA tests

Depending on who is testing FPGA, we can set two different aims for test: full check of FPGA and partial check of part of FPGA, what is used for function realization. The manufacturers are interested in the first aim, when users are interested in the second. Manufacturer must check every cell, every connection and so on. Manufacturing-Oriented tests are divided in few tasks for different components testing: Configurable Logic Blocks (CLB) [4,5]; interconnections between CLB, interconnections network [6]; Look-Up Tables (LUT) [7]. When all such procedures passes the manufacturer test is completed. 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 users are interested in good functionality only of PLD's part, witch need for application, after they have programmed it. For this aim 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 [8,9]. 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.

Architecture of FPGA

Usually each PLD's family separates from others by some features, but almost all of them consist of a matrix of CLB (configurable logic blocks) and CIOB (configurable blocks of inputs and outputs). All CLB and CIOB are connected to each other by CBI (configurable blocks of interconnections) and a lot of conductors (Fig. 1.). 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. 2.). 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.

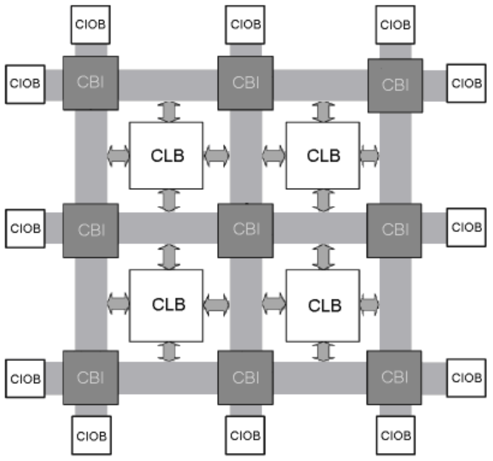


Fig. 1. Simplified architecture of PLD

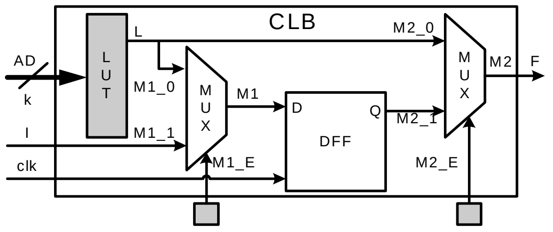


Fig. 2. Simplified architecture of CLB

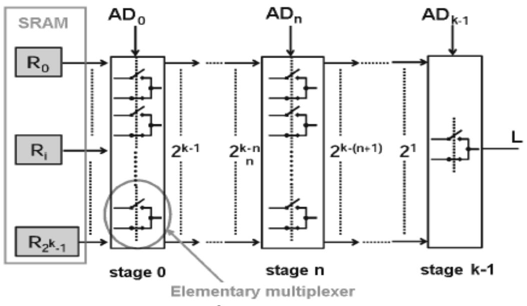


Fig. 3. Common structure of LUT

Grey boxes in Fig. 2 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 2^k bit's of RAM. In this paper the RAM mode is not considered. 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. 3 [10]. R_0 – R_{2^k-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 R_i can be connected to the output L. The number of stages and multiplexers in the stage depends on how many LUT have inputs.

Almost all SRAM based FPGAs have most same architecture and internal CLB and LUT structure, like it is showed in Fig. 1, 2 and 3. Such FPGAs are Virtex–4 [11], ProASIC™ [12], ispXPGA [13], Stratix II [14] and others. They vary in number of IO, CLB, LUT's inputs and flip flops in one CLB.

The transition fault model of FPGA

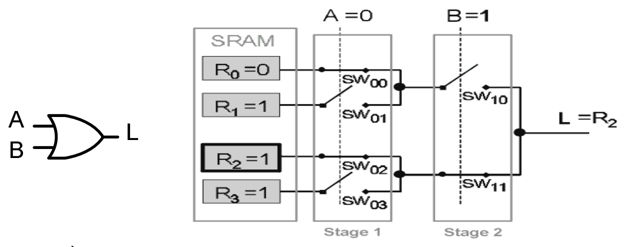
It is usual to model the functionality of electronic devices, but it is possible to model faults too. To do it we need special models of real devices. Using them it is possible to make tests for real devices. Most popular are stuck-at, path delay (transition) and element delay fault models. The fault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

Realistically, defects can be divided by derivation to processing defects (bed contact, parasitic transistors, disruption of oxide), defects of silicon (fracture, crystal's unsolid and amorphous surface, migration of ions), time depending faults (disrupted dielectric, leak of current), packaging (damage of contacts and/or isolation). 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 zero to one (from one to zero, in transition to zero case) does not effects any output or trigger of the circuit in particular set time period, when it begins counting from transition's lead-off. There is only one difference between transition and delay faults: particular set time range for delay fault for transition. And in case of transition fault it is took that time for transition is infinite. Transition and delay faults can appear for few reasons: some manufacture process failure (resistance and capacity exceeds projected) and harmful environmental impact, overheat or too low environmental temperature of exploitation (physical characteristics of crystal are changed).

There are couple of vectors (V1, V2) used for transition faults test. They can't be equal. Also all R meanings (Fig. 3) 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 [15]. V1 is a vector for initiation, which sets testable node to appropriate logical value (0 for STR and 1 for STF). 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 (paths). Then rising front could check transition to logical one on the concrete path, and falling front could check transition to logical zero faults on the specific path. Unfortunately, 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 (0 or 1). So the transition can't be followed in the circuit's primary outputs or scanning triggers.

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 PLLs. Two small circuits could be taken as an example. There is one OR element (Fig. 4a) in one circuit and there is LUT of the same function in the other (Fig. 4b). First one has only 6 possible transition faults. It is two STR and two STF on the inputs A and B, and one STR and one STF faults in the output L. It is obvious, that second (Fig. 4b) circuit differs from the first. It is enough one LUT with two inputs to realize OR function. Such two inputs LUT will have three multiplexers ($SW_{00\text{-}01}$ $SW_{02\text{-}03}$ and $SW_{10\text{-}11}$) and all of them will be connected into two stages. So we will have already tree elements and all off them have tree inputs and one output, so it will be 8 possible faults per component and total 24. Half of them are STR and half – STF. This simple example clearly shows how different are these two circuits in low level schematic. Possible faults are also different. So why tests generated for one schematic using traditional methods and models will fail, when it will be used for other schematic. Such test will check only part of possible faults of the circuit with same functionality and programmed in PLD. It was proved by experiments and published earlier [2].



a) **b)**

Fig. 4. a) Simple OR element; b) LUT with OR function

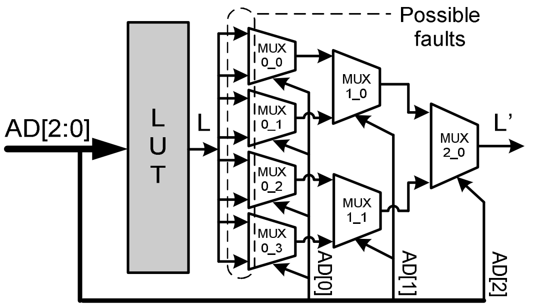


Fig. 5. Model of the LUT

The transition fault model of LUT

The fault model can be used to get exhaustive test for some configuration. Such model must fully represent the functionality and faults of LUT. When we are making tests for PLD's circuits, we are facing with few main problems:

1. Testing vectors generated of ASIC's circuits don't test PLL's fully.
2. Circuits synthesized using libraries of PLL elements are hardly testable. ATPG are overloaded with huge amount of untestable faults and it takes big amount of time.
3. After synthesis it is known only what must be loaded into LUTs, but ATPG doesn't know the LUT's structure.

For these problems it is proposed the model for transition faults for FPGA's (Fig. 5). There the model of LUT realizes some function with three operands. There is multiplexers group inserted in the output of LUT. This group represents the internal structure of real LUT. So, the internal structure of LUT is showed to ATPG. Now we have got model representing the real LUT.

To increase speed of test generation it is possible to limit set of faults with one condition – limitations can't reduce the quality of test. Test patterns initiate wave of transitions in whole path from initiation point to primary output or scanning trigger. Based on this we can reduce the set of all possible faults to faults only in the inputs of first stage of multiplexers (MUX0_0, MUX0_1, MUX0_2, MUX0_3). These inputs represent all possible paths in the LUT and such limitations don't effect test quality. It was checked by experiments. There is only one possible transition fault for one input – STF or STR; it depends on value R_n. We'll list only these possible faults, so we'll be sure that ATPG will check all possible paths if only they aren't limited by function. We'll use the quantity of checked paths as the gauge of quality of the generated test.

Such model can be used in this way:

- Circuit must be synthesized using PLD elements. After synthesis we'll get to know what CLB will be used and which function implemented using specific LUT.
- The new net list must be saved writing not the table of truth, but structure of elements. There is important only the same function, it will be used as black box in ATPG.
- The groups of multiplexers must be inserted in the outputs of LUTs. The size of group depends on number of inputs of specific LUT.
- The list of possible faults must be limited as it is described in the model.
- Some ATPG must be used to generate test patterns. Such test will be the exhaustive and able to check more possible faults of PLD.

Experimental results

ISCAS-85 benchmarks [16] and Synopsys software "Design Analyzer" and "Tetra Max" [17] were used for experiments. The results of experiments with tests for transition faults are showed in the Table 1. The objectives of experiment were: compare quality of traditional ASIC and exhaustive PLD tests; prove that tests generated for simple ASIC circuits can not fully check circuits synthesized in some PLD library of elements.

Table 1. Exhaustivity of tests of ISCAS-85 circuits

Circuit	Exhaustivity of test		
	Test for Class	Test for Virtex	Test for model
C17	52,08%	54,16%	70,83%
C432	55,85%	54,03%	81,35%
C499	41,16%	42,29%	51,35%
C880	55,97%	56,72%	71,1%
C1355	40,62%	41,12%	49,33%
C1908	47,45%	45,69%	58,83%
C2670	50,05%	50,67%	60,73%
C3540	54,48%	53,92%	65,88%
C5315	56,76%	56,93%	62,07%
C6288	44,68%	50,2%	52,38%
C7552	54,78%	54,74%	59,74%

The experiments were made in this way:

1. There are generated three different tests for transition faults for circuits synthesized using Class (ASIC), Virtex (PLD) and for described model.
2. All three tests compared to one criterion – how many testable transition faults on the inputs of multiplexers in the first stage of model they can check.

Tests generated for model are more exhaustive that tests generated for Class circuits, as for Virtex circuits too. The 1st table shows that. And there is a bigger probability what more exhaustive test will check PLD better.

Conclusions

1. There was presented one model for exhaustive test generation for transition faults of SRAM based PLDs. The need of such model was proved by experiments.
2. Presented model can be used for testing of PLDs. The traditional ATPG software for ASIC circuits can be used for PLDs through this model.
3. Tests generated using model are more exhaustive then tests generated using traditional methods and can check PLD better.

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. **Abramovici M., Stroud C.** BIST-Based Test and Diagnosis of FPGA Logic Blocks // IEEE Transactions on VLSI Systems. – 2001. – No. 9–1. – P. 159–172.
5. **Zhao L., Walker D.M., Lombardi F.** Detection of Bridging Faults in Logic Resources of Configurable FPGAs Using Iddq // International Test Conference. – 1998. – P. 1037–1046.
6. **Renovell M., Portal J. P., Figueras J., Zorian Y.** Testing the Interconnect of RAM-Based FPGAs // IEEE Design & Test of Computers. – 1998. – P. 45–50.
7. **Renovell M., Portal J. M., Figueras J., Zorian Y.** SRAM-Based FPGA's: Testing the LUT/RAM Modules // IEEE International Test Conference. – 1998. – P. 1102–1111.
8. **Šeinauskas R.** ASIC Design Flow and Test Generation Capabilities // Information technology and control. – 2003. – No. 1(26). – P. 55–60.
9. **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.
10. **Girard P., H´eron O., Pravossoudovitch S., Renovell M.** Delay Fault Testing of Look-Up Tables in SRAM-Based FPGAs // Journal Of Electronic testing: Theory and Applications. – 2005. – No. 21. – P. 43–55.
11. **Xilinx Inc.** Virtex–4 Overview. – 2005. – http://www.xilinx.com/products/silicon_solutions/fpgas/virtex/virtex4/overview/index.htm .
12. **Actel Corporation.** ProASIC – The Nonvolatile Reprogrammable Gate Array. – 2004. – http://www.actel.com/products/proasic/index.html .
13. **Lattice Semiconductor Co.** Instant-On FPGA Solutions: ispXPGA. – 2006. –http://www.latticesemi.com/products/fpga/ispxpga/index.cfm .
14. **Altera Co.** Stratix II Devices: The Biggest & Fastest FPGAs. – 2006. – http://www.altera.com/products/devices/stratix2/st2-index.jsp .
15. **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.
16. **Collaborative Benchmarking Laboratory.** ISCAS'85 Benchmark Information. – 1997. – http://www.cbl.ncsu.edu/CBL_Docs/iscas85.html .
17. **Synopsys Inc.** Test Automation. – 2005. – http://www.synopsys.com/products/solutions/galaxy/test/test.html.

Submitted for publication 2006 03 13

V. Abraitis, Ž. Tamoševičius. The Transition Fault Model of Programmable Logic // Electronics and Electrical Engineering. – Kaunas: Technologija, 2008. – No. 1(81). – P. 73–76.

There is presented the fault model of programmable integrated circuits in this paper, when programmable integrated circuits are configured to implement a given application. Proposed fault model can be used with traditionally automatic test sequence generators and result will be exhaustive test of transition faults for programmable integrated circuits with given configuration. Model was tested using Virtex family PFGAs. Ill. 5, bibl. 17 (in English; summaries in English, Russian and Lithuanian).

В. Абрайтис, Ж. Тамошявичюс. Модель неисправностей переключения программируемых интегральных схем // Электроника и электротехника. – Каунас: Технология, 2008. – № 1(81). – С. 73–76.

Рассматриваются модели неисправностей переключения элементов программируемых интегральных схем (ИС), когда программируемые интегральные схемы запрограммированы выполнять данную функцию. Предлагаемые модели неисправностей переключения могут быть использованы с традиционными генераторами тестовых последовательностей и результатом будет исчерпывающий тест для программируемой интегральной схемы с данной конфигурацией. Модели были проверены, используя программируемые интегральные схемы семейства Virtex. Ил. 5, библ. 17 (на английском языке; рефераты на английском, русском и литовском яз.).

V. Abraitis, Ž. Tamoševičius. Programuojamųjų loginių lustų perjungimo gedimų modelis // Elektronika ir elektrotechnika. – Kaunas: Technologija, 2008. – Nr. 1(81). – P. 73–76.

Pateikiamas programuojamosios logikos elemento perjungimo gedimų modelis, leidžiantis naudoti tradicinius testinių sekų generatorius programuojamiesiems lustams tikrinti. Pateiktu modeliu tradicinis automatinis testinių sekų generatorius priverčiamas perrinkti visas į elemento įėjimus patenkančių rinkinių kombinacijas, apribojamas tik elemento funkcijų. Prieš gaunant testinius rinkinius, formuojamas tik patikrinamų gedimų sąrašas ir šitaip palengvinamas ir paspartinamas testinių sekų generatoriaus darbas. Atliktas eksperimentinis modelio patikrinimas naudojant Virtex šeimos programuojamuosius lustus. Il. 5, bibl. 17 (anglų kalba; santraukos anglų, rusų ir lietuvių k.).