設計自動化 91－2
(1999. 2. 4)

あらまし

キーワード

FPGAs Complete Fault Diagnosis Based on Binary Tree BIST Method

Abderrahim Doumar[†] Toshiaki Ohmameuda[††] Hideo Ito[††]

[†] Graduate School of Science and Technology, Chiba University
[††] Faculty of Engineering, Chiba University
1-33, Yayoi-cho, Inage-ku, Chiba 263-8522, Japan

E-MAIL:{doumar,mame,ito}@icsd2.tj.chiba-u.ac.jp

Abstract This paper presents a new approach for test and diagnosis of faults in field programmable gate arrays (FPGAs). The new method exploits the configurability and the programmability of SRAM-based FPGAs and implements connections between the configurable logic blocks (CLBs) as a binary tree. The proposed scheme is based on BIST (built-in-self-testing) method, and the implementation does not need any hardware overhead. It is proved that this approach detects the multiple faults and locates single faults. The method is also able to give the exact locations of one part of multiple faults, while it gives some possible locations for other multiple faults. The simulation results indicate that the proposed method covers 100% of the modelled faults .

Key words FPGA, Testing, Configurability, Binary tree, BIST, Diagnosis.

1 Introduction

Field programmable gate array (FPGA) is widely used since it can be programmed to implement any logic circuit. Different architectures of FPGAs are presented by different manufacturers such as Xlinix [23] or Altera [24]. The SRAM-based FPGA, which is the most famous type, usually consists of configurable logic blocks (CLBs) connected by interconnection network, and programmable input outputs (I/O) blocks.

The test of an FPGAs chip has recently attracted the interest of many researches [1] - [20]. The research on the test of FPGA can be classified globaly in some categories. The first category [21] , [22] takes advantage of FPGA reprogrammabitity by treating a test just as another application to be implemented in the FPGA. Unfortunately it has low faults coverage rate. While the second category [1] - [20] exploits the regular array structure of an FPGA by configuring it as one or more iterative logic arrays. However it locates, with difficulty, a single fault and can not directely locate multiple faults. Other cathegories test only one part from FPGA, for exemple interconnection network [9] - [11].

In this paper a novel approach for testing and locating faults in SRAM-based FPGAs is proposed. It detects and locates 100% of single faults and locates, under some conditions, all multiple faults. In the cases of non-satisfaction of these conditions this method indicates some possible locations for multiple faults. IN addition a majority of faults in interconnection network can be detected.

In next section a background and the basic structure of BIST method are described. In section 3 the binary tree architecture (new scheme) is presented and discussed. Morover this section presents a general concept, a diagnosis and the approach limitations. Simulation results and evaluation of this method are studied in section 4. Then a conclusion makes end to this study.

2 Background

In this section the model of FPGA which is under test and the outline of the previous works of FPGA testing [1] - [4] based on BIST method are presented.

2.1 Basic SRAM-FPGA Model

FPGA consists of an array of N × N CLBs which can be programmed with configuration cells to generate logical functions (Figure 1). The set of all programming bits establishes a configuration. Every CLB can be connected to other CLBs using the interconnection network. The final results can be transmitted to the outside of FPGA using the inputs/outputs blocks.

Different FPGAs types are present in the market with several usable modes. For example a configurable logic blocks in some FPGAs (XC 4000, ORCA,$\cdots$,ect) have a possibility to be programmed as RAM (RAM mode). We focus on the general FPGA structure which it CLBs implement either a normal logical function or RAM.

2.2 BIST Method

Earlier work proposed the application of BIST for FPGA [1]. As shown in Figure 2, the set of all CLBs in FPGAs was divided into 2 parts. The CLBs in the first part are configured to implement a test pattern generator (TPG) and output response analyzers(ORAs), while the CLBs in the second part are used as blocks under test (BUTs). Since the two parts can not be tested simultaneously, the method proposed in [1] has at least two sessions for testing all CLBs in FPGAs :

1. Session 1: The CLBs of part 1 are under test and CLBs of part 2 are used in the implementation of TPG and ORAs.
2. Session 2: The CLBs of part 2 are under test and CLBs of part 1 are used in the implementation of TPG and ORAs.

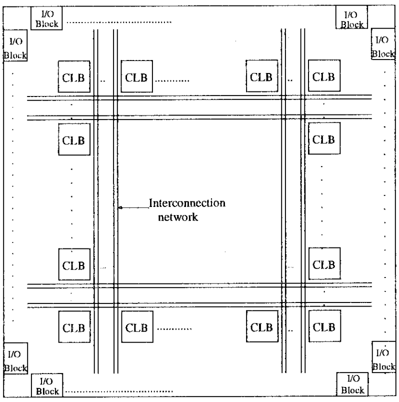

Figure 1: Basic structure of FPGAs.

For a complete fault coverage, every CLB must be under test at least in one session. Since the ORAs and TPG are implemented inside FPGA, the only cost of the BIST method is the additional memory for storing the data required to reconfigure the FPGA. This architecture needs an extreme flexibility in an interconnection network resources because every BUT is fully connected from one side to TPG and from other side to ORA and all BUTs are laid out within 2 dimensional array.

In one solutions for this problem BUTs in part 2 are arranged, by cascading its in different rows[3],[4]. Every row is called "iterative logic array (ILA)". The ILA connects the outputs of CLBs (except the last one which is connected to the ORA) to the next CLB inputs (except the first one witch is connected to the TPG) in the ILA. Since the number of inputs is smaller than the number of outputs, the authors of [3], [4] used helpers to offer for CLBs in ILAs the necessary inputs as shown in (Figure 3). The scheme proposed by C.Stroud implements each helper at one CLB. The exact helper's functions depend on the FPGAs type and modes which are defined by FPGAs data books.

3 Notations and Definitions

LUTs : Look-up tables

N_{ORA} : The number of ORAs implemented in FPGA in each session.

$N_{BUT}(i)$: The number of CLBs tested in FPGA in the session i.

N_{TPG} : The number of CLBs needed to implement TPG in each session.

P: The number of levels in binary tree. For example P=4 in Figure 10.

$N_{I/O}$: The number of input/output blocks in FPGA.

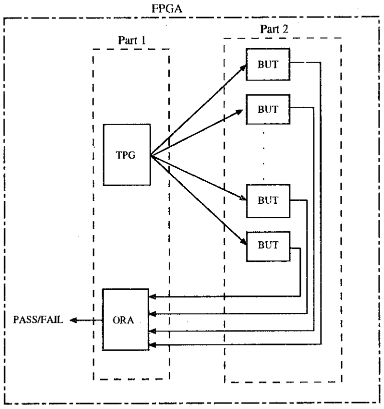

Figure 2: BIST scheme for FPGA.

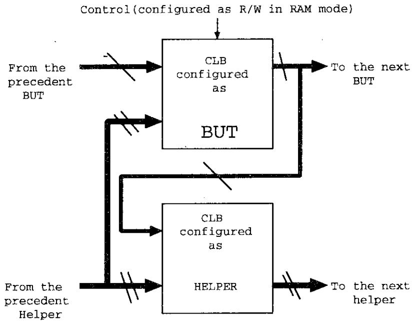

Figure 3: One BUT connection from ILA BIST structure.

N_{HELPER} : The number of helpers needed in each session.

N_{BUTL} : The number of BUTs at the last level in the binary-tree.

N: The number of CLB rows in FPGA.

M: The number of CLB coulombs in FPGA.

Q: The number of CLBs needed to implement one ORA.

$N_{BUT}(i)$: The number of BUTs used at session (i).

$RTS(i)$: The response of test sequence i. In Figure 11 $RTS(i)$ is the value of RTS during the test sequence i.

Complete binary tree : A binary tree is called complete when every BUT (except the BUTs in the last level in the tree) is connected to two BUTs. In the other case the tree is called incomplete.

Phase : A phase consists of programming an FPGA with one configuration followed by test vector application. The time reserved to implement one phase is very large than the time required by the application of the test sequence because all configuration cells must be loaded from the configuration program.

Subcircuit : A CLB is divided into some parts (subcircuits) $\{SUB_1 \cdots SUB_n\}$, for example every multiplexer, flip flop and LUT are considered as subcircuits. CLB is constructed by the union of all subcircuits (CLB = $\bigcup_{i=1}^{i=n} SUB_i$).

C-testable : An FPGA is C-testable if the testing time is independent of the number of CLBs [15],[16],[17]. In particular if the CLBs are connected within binary tree it is independent of the tree size.

4 Binary Tree New Architecture

An illustration example of the basic concept is presented first, the fault model assumed in this approach is defined next, the binary tree architecture presentation follows, and the test conditions are outlined at the end of this section.

4.1 Preliminaries

For simplicity of illustration, let's consider an example of 6 BUTs. Testing the CLBs one by one will take certainly a long time. While connecting the CLBs in an array(Figure 4) can resolve this problem, however it is not able to locate faults, because CLBs (except the last one) outputs are not directly observable. If we decide to put all CLBs outputs observable, CLBs must be connected as shown in Figure 5, but every CLB needs one I/O block at FPGA. This condition is not always satisfied.

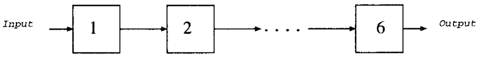

Figure 4: Example of 6 CLBs in an array.

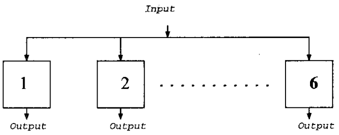

Figure 5: Example of 6 CLBs with a direct observability.

Connecting CLBs as a binary tree as shown in Figure 6 reduces the number of I/Os blocks needed to the half (in this example 6 CLBs are presented, 4 CLBs are directly observed while 2 CLBs doesn't need the direct observability).

Unfortunately the number of CLB inputs and outputs are different. In addition each of CLB3, CLB4, CLB5 and CLB6 needs some suplementary inputs which can be driven from other CLBs configured as helpers as shown in Figure 7. The helper function in this Figure is a simple multiplexer and the missing inputs signals for CLB3, CLB4, CLB5 and CLB6 are driven from the helper input to its output.

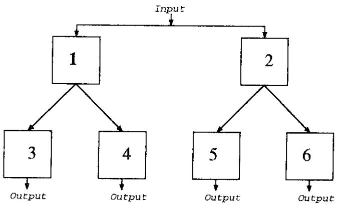

Figure 6: Example of 6 CLBs in binary tree.

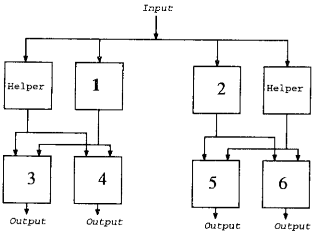

Figure 7: Example of 6 CLBs in binary tree with helpers.

4.2 Fault Model

The basic internal architecture of CLB, shown in Figure 8, is built by three components: lookup tables, multiplexers and D-flip flops. The LUT implements either a logical function with k1 inputs or RAM. This model generalizes all conventional CLBs.

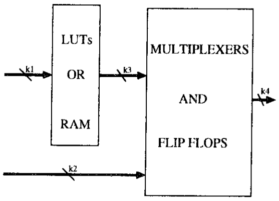

Figure 8: Basic internal CLB architecture.

A fault can affect any of these components. And it is modelled as follows:

For an LUT, a stuck at fault is assumed in the LUT's inputs and outputs. A fault can also affect any memory cell sothat the right data can not be seen at the LUT's output. The decoder which makes it incapable to generate the right addresses.

For a multiplexer, a functional fault model is assumed. The functional model is enough because the aim is to be sure that multiplexer realizes the right functions. The stuck at fault model is not assumed in this part because the internal structure of multiplexers varies from one company to another.

For a D-flip-flop, a functional fault model is assumed. A fault can make it incapable to storage data or unable to set or reset.

Table 1: Test Sessions Proposed by the Binary Tree Method .

	Set1	Set2	Set3
Session 1	BUTs	TPG,ORAs	OR Helpers
Session 2	TPG,ORAs	Helpers	BUTs
Session 3	Helpers	BUTs	TPG,ORAs

For the CLB input connections or the CLB output connections or any connection between multiplexers and multiplexers or multiplexers and flip/flops a fault is assumed to be "stuck at".

The proposed method detects all faults in CLBs. However, even if a big part of interconnection network faults can be detected, this possibility is not cosidered. I/Os blocks of FPGAs are assumed fault free. This condition is normal since the I/Os blocks can be separately tested with boundary scan [23] .

4.3 Test Architecture

This study proposes a new testing implementation of FPGAs based on binary tree. As shown in Figure 9, CLBs are divided into three sets: The first set is used as BUTs and it is connected as binary tree while the second set is used to construct TPGs and ORAs, and the third set is used to implement helpers. Each BUT in the tree needs one helper to provide it with the necessary inputs. Thus the number of CLBs implementing helpers is equal to a number of CLBs in the tree. The helper function can be defined at each mode. For example for RAM mode the helpers can be implemented as multiplexers. For more clearness, the helpers are not represented in Figure 10. Every BUT have different name in this Figure. For exemple the BUT connected to BUT "A" and BUT "B" is named "AB". This notation will serve in locating BUTs in the tree.

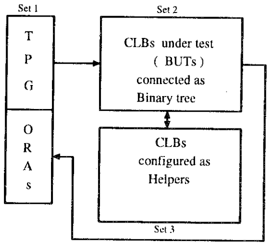

Figure 9: The Basic architecture for the Binary Tree Method.

The proposed test needs 3 sessions (Table 1), in each session one set will be tested :

Following the fact that in this test all BUTs in the tree have the same inputs value(some inputs are conducted from TPG block to the designed CLB across helpers and other part are conducted by CLBs itselfs), in absence of faults, the outputs must be the same. An ORA will have two inputs series (Figure 11): The first

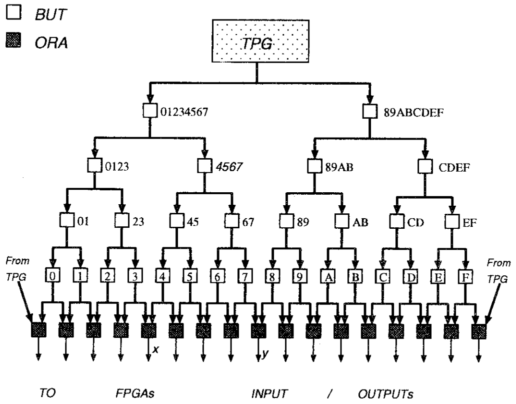

Figure 10: Binary tree structure.

inputs series represent BUT "i" outputs and the second inputs series represent BUT "j" outputs. Each input signal in series 1 will be compared with the corresponding one in the second series inputs and the final result is given by the signal RTS. If for one test vector, an ORA output is equal to "1" , it will not change for all the following test vectors . A comparator can be either an analyzer or a comparator-based analyzer. But in general case its architecture depends on the number of inputs and the number of outputs of CLBs. The condition of C-testability are fully satisfied, since all BUTs in the tree have the same input in absence of faults.

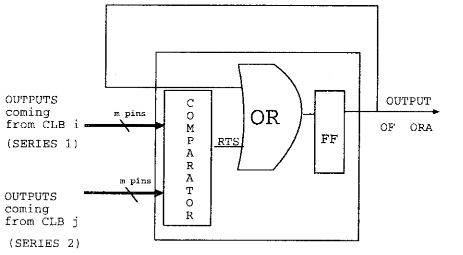

Figure 11: Basic ORA structure.

4.4 Test Conditions

Since the number of input/outputs pins in FPGA is limited and the testing time must be the as short as possible, some conditions must be satisfied.

The test is completely accomplished at three sessions, thus the following expression must be satisfied.

$$N_{BUT}(1) + N_{BUT}(2) + N_{BUT}(3) \geq N \times M. \quad (1)$$

Let's take :

$$N_{BUT} = Min(N_{BUT}(1), N_{BUT}(2), N_{BUT}(3)). \quad (2)$$

In this case, the following expression is sufficient to assure that all CLBs are tested in three session.

$$3 \times N_{BUT} \geq N \times M. \quad (3)$$

On the other hand the ORAs outputs must be observed from outside FPGA. Therefore, each ORA output should be connected to one I/O block. Consequently the number of FPGAs I/O blocks is larger than that of the ORAs :

$$N_{I/O} \geq N_{ORA}. \quad (4)$$

Referring to the binary tree structure, easily the following deduction is done :

$$N_{BUTL} = 2^P, \quad (5)$$

and

$$N_{ORA} = N_{BUTL} + 1. \quad (6)$$

Consequently

$$N_{I/O} \geq 2^P + 1. \quad (7)$$

Since the ORAs, TPGs, helpers and BUTs are implemented on the same FPGA, the following expression holds :

$$M \times N \geq Q \times N_{ORA} + N_{BUT} + N_{HEPER} + N_{TPG}. \quad (8)$$

The structure of binary tree shows easily that the number of BUTs in each session is:

$$N_{BUT} = 2^{P+1} - 2. \quad (9)$$

In addition the number helpers is equal to a number of BUTs:

$$N_{HELPER} = N_{BUT}. \quad (10)$$

Since the TPGs and ORAs implementations depend on a BUT structure, a general expression of N_{TPG} and N_{ORA} can not be expressed using only the parameter P.

The above discussion can be summarized in three conditions:

$$Log_2(N_{I/O} - 1) \geq P. \quad (11)$$

$$P \geq Log_2(N \times M/3 + 2) - 1. \quad (12)$$

$$Log_2((N \times M - Q - N_{TPG} + 4)/(Q + 4)) \geq P. \quad (13)$$

In the most cases, the three conditions are satisfied by adequate choice of P. But if theses conditions are not satisfied simultaneously, the ORAs must be designed in new structure which is not expensive in term of CLBs, or add some BUTs in the binary-tree. For example in Figure 10, some BUTs can be added between BUT"EF" and BUT "F". However, this solution will make a confusion between BUTs during a fault localization.

If the binary tree is not complete the same kind of study can be done. But the approach performance will be less in term of localization.

5 Test and Diagnosis of Faults

In this section a single fault detection and localization is discussed first. Then the disscution of multiple faults detection and localization follows. The aim of this section is detecting and locating faulty CLBs. Unfortunately the localization of faulty subcircuits is not considered.

5.1 Single Fault Detection and Localization

Let's take P_{CLB} the number of phases to test exaustively the CLB. If a fault happens at one CLB, the concerned CLB will be declared faulty and then it will be detected and located by the proposed diagnosis after P_{CLB} phases. A single fault detection is assured, since all BUTs outputs (except the CLBs of the last stage which are connected to the ORAs inputs) are connected to two next BUTs. For example, if BUT "AB" (Figure 10) is faulty, this error will be transmeted to the following BUTs "A" and "B" and it will be immediately seen at the outputs "x" and "y".

Every single fault generates one different ORAs output vector. Therefore a single fault localization is assured.

5.2 Multiple Fault Detection and Localization

A multiple faults are also detected for the same reasons. More difficult task is the localization of multiple faults, because logically, $2^{(2^{P+1}-2)}$ different multiple faults can happen, while only $2^{(2^{P}+1)}$ different outputs are generated by ORAs. Therefore the maximum multiple faults which can be located is $2^{(2^{P}+1)}$. However this number is smaller in practice.

This method consist of testing every subcircuit with one test sequence specially built for it. Then two different subciruits have two different test sequences.

Assumption 1 Multiple faulty BUTs can happen only if the faulty subcircuits in the faulty BUTs are different.

Theorem 1 With the assumption 1, an ORAs outputs of any multiple faults is only "OR" of the corresponding ORAs outputs of single faults.

Proof : Since multiple faults happen at different subcircuits, a test pattern for theses subcircuits are different. Let's B=$\{SUB_1, \cdots, SUB_n\}$ be the set of subcircuits in one CLB and TS=$\{TS_1, \cdots, TS_n\}$ a set of corresponding test sequences. This test consist of an application sequentially from TS_1 to TS_n. For every test sequence i, at each ORA one $RTS(i)$ is decided. Following the ORAs structure (Figure 11), ORA output for a $testsequence(i+1) = (RTS(i+1))OR(RTS(i))$, where the first ORA output for a $testsequence(0) = 0$ (initialization). Concequentely :

ORA output = (RTS(1)) OR (RTS(2)) OR $\cdots$ OR (RTS(n)),

where "ORA output" is an ORA output after an application of all test sequences.

Obviously, if the number of subcircuits is smaller than the number of multiple faults, at least two BUTs have the same faulty subcircuits, therefore faults can be recovered. The number of subcircuits must be larger than multiple faults.

With the information concerning the length of multiple faults (number of faulty CLBs) and the stage which encompass faults, we can in most cases locate precisely faults, but without this information, for each detection of multiple faults one number of fault possibilities are considered : For example in Figure 6 if CLB6 and CLB2 are faulty, we have 4 cases possible considered by a fault localization :

Possibility 1 : CLB2 and CLB6 are faulty.

Possibility 2 : CLB5 and CLB6 are faulty.

Possibility 3 : CLB2 and CLB5 and CLB6 are faulty.

Possibility 4 : CLB2 and CLB5 are faulty.

Assumption 2 Multiple faulty CLBs can happen only if the fauty CLBs have the same faulty subcircuits. Consequentely the faults are concerning the same subcircuits in these CLBs.

Theorem 2 With the assumption 2, the ORAs outputs of any multiple faults in one stage is only "XOR" of corresponding outputs of singles faults.

Proof : The proof is evident and it has the same consequences as before. In this case only faults in the same stage are considered because the corresponding outputs of two faults in one branch depends enormously on the test patterns.

6 Simulation and Evaluation

The proposed method can be applied to any FPGA (XC3000,XC4000,XC5200 $\cdots$ etc). This section illustrates the application for XC4000 family because its LUTs have a possibility to be programmed as RAM as well as a normal combinational circuit. The simulation results obtained by testing a single CLB are presented first followed by a comparision of this method with the other principal methods.

6.1 Simulation Results

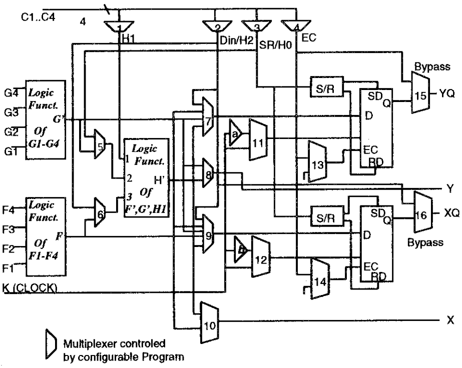

Figure 12: Basic structure of XC4000 CLB.

The XC4000 CLB is constructed by the following components, as shown in Figure 12 :

- Six 4-inputs LUTs and one 3-inputs LUT.
- Two flip-flops.
- Six multiplexers with 4 inputs and one output (multiplexers:1, 2, 3, 4, 7 and 9).
- Ten multiplexers with 2 inputs and one outputs (multiplexers:5, 6, 8, 10, 11, 12, 13, 14, 15 and 16).
- Two "NOT" gates and two CONTROL S/R blocks which can be modeled by a 2 input-multiplexers.

A CLB has 13 inputs $\{F1, \cdots, F4, G1, \cdots, G4, C1, \cdots, C4, CLOCK(K)\}$, 4 outputs {YQ,XQ,X,Y} and Table 2 shows the reserved number of configuration cells for every subcircuits. Finally, 60 configuration cells and 12 inputs (Clock and associated subcircuits are not counted) are present. Theoretically for the complete exhaustive test 2^{60+12} test vectors must be submitted to each CLB.

For a complete test achievement of XC4000 (all modes) in minimum test time possible, only 10 phases with 76 test vectors are envisaged. And 1604 faults are generated in this fault simulation with a single CLB. At every phase one part from these faults is detected and then a fault coverage is determined.

The Table 3 shows that at the end of the last phase, 100% of fault coverage is obtained.

6.2 Evaluation

This approach is evaluated and compared with the principal previous approaches: BIST approach [1]-[4], XOR based approach [8], AND/OR tree approach [7], Naive approach [6] and Array-based approach [6]. The same strategy of evalution as in [5] [7] [8] will be followed. The comparison is done with respect to the following issues:

- Generality: Which is defined as the ability of the FPGA model -assumed in this method- to include all conventional FPGA without any structural problems such as I/Os limitations.

- Test time : Which can be evaluated by the total number of phases in all sessions.

- Single fault diagnosis ability: Which is defined as the ability of the method to locate single faults.

- Multiple fault diagnosis ability: Which is defined as the ability of the method to locate multiple faults.

In (Table 4) the following conventions are used:

++: Means widely better than the method in the reference.

+: Means better than the method taken as reference.

-: Means less than the method taken as reference.

=: Means same as the method taken as reference.

The Naive approach is used as a reference for this comparaison (Table 4).

The binary tree method has a good points regarding the single fault diagnosis and multiple fault diagnosis, but it has bad point regarding the test time.

Table 2: Number of BUT configuration cells.

categories	A number of units	A configuration cells by unit	A total number
4-inputs LUT	2	16	32
3-inputs LUT	1	8	8
4-input Multiplexers	6	2	12
2-input Multiplexers	8	1	8

7 Conclusion

In this paper a novel approach for test and diagnosis SRAM-based FPGAs is presented. The proposed approach is based on a binary-tree. It detects and locates a single fault, detects multiple faults and it is able to give the exact locations of one part of multiple faults, while it gives some possible locations for others multiple faults. The binary tree method is applied to CLBs with different internal architecture. The simulation results indicate that the proposed method covers 100% modelled faults.

References

[1] C.Stroud, S.Konala, P.Chen and M.Abramovici, "BUILT-in self-test of logics Blocks in FPGAs", 14th VLSI test symposium, pp. 387-392, 1996.

[2] C.Stroud, P.Chen,S.Konala and M.Abramovici, "Evaluation of FPGAs resources for BUILT-in self-test of programmable logic Blocks", PROC. 1996 ACM/SIGDA International. symp. on FPGAs, pp. 107-113, Feb. 1996.

[3] C.Stroud,E.Lee, S.Konala and M.Abramovici, "Using ILA test for BIST in FPGAs", International test conference IEEE, pp. 68-75, 1996.

[4] C.Stroud,E.Lee, S.Konala and M.Abramovici, "BIST-based diagnostics of FPGAs logic blocks", International test conference, pp. 539-447, 1996.

[5] W.K.Huang, F.J.Meyer and F.Lombardi,"Testing Configurable LUT-Based FPGA's", IEEE trans on VLSI, VOL.6, NO.2, JUNE 1998.

[6] W.K.Huang and F.Lombardi, "A general technique for testing FPGAs", IEEE VLSI test symposium, priceton NJ, p 450-455, 1996

[7] W.K.Huang, F.J.Meyer and F.Lombardi, "Multiple fault detection in logic resources of FPGAs", DFT-VLSI, pp 186-194, 1997.

[8] W.K.Huang, F.J.Meyer and F.Lombardi, "A XOR-tree based technique for constant testability of configurable FPGAs", ATS, pp. 248-253, 1997.

[9] F.Hanchekand and S.Dutt, "Methodologies for tolerating cell and interconnect faults in FPGAs", IEEE transactions, Vol, 47, no, 1, pp.15-33, January 1998.

[10] T.Liu and F.Lombardi, "On soft switch programming for reconfigurable array systems", DFT-VLSI, pp. 203-211, 1994.

[11] M.Renovell et al, "Testing the interconnect of RAM-based FPGAs", IEEE Design and test of computers, pp. 45-50,1998.

[12] T.Inoue, S.Miyazaki and H.Fujiwara, "Universal fault diagnosis for lookup table FPGAs", IEEE Design and test of computers, pp. 39-44, 1998.

[13] T.Inoue, S.Miyazaki and H.Fujiwara, "On the complexity of universal fault Diagnosis for lookup table FPGAs", IEEE Design and test of computers, pp. 276-281, 1997.

[14] M.Renovell et al,"Test pattern and test configuration generation methodology for the logic of RAM-based FPGAs, ATS, pp. 254-259, 1997.

[15] H.Elhuni and al,"C-Testability of two-Dimensional iterative arrays", IEEE transaction on computer-aided design, vol.cad-5,no.4, pp. 573-581, 1986.

Table 3: Configuration phases.

Phases	Subcircuit Tested	Number of Test vectors	Number of Faults Detected	Fault Coverage
1	RAM AND Flips-Flpops	16	1212	75%
2	4-inputs LUT And FFs	16	194	87,6%
3	4-inputs LUT And FFs	16	132	95,8%
4	MUXs	2	3	96.1 %
5	MUXs	2	4	96.3%
6	MUXs	2	3	96,5 %
7	MUXs	2	3	96.7 %
8	3-inputs LUT	8	18	97.8 %
9	3-inputs LUT	8	15	98.7 %
10	carry	4	35	100 %

Table 4: Approaches Evaluation.

	BIST	Naive	Array-based	AND/OR tree	XOR tree	Binary tree
Generality	-	Reference	-	+	+	=
Test time	-	Reference	++	+	++	-
Single fault diagnosis	able	unable	unable	unable	unable	able
Multiple fault diagnosis	unable	unable	unable	unable	unable	able

[16] A.D.Friedman, "Easily testable iterative systems", IEEE transaction on computers, vol. c-22, no.12, pp 1061-1064, 1973.

[17] P.R.Menon and A.Friendman, " Fault detection in iterative logic arrays", IEEE transactions on computers, vol.c-20, no.5, pp.524-535,1971

[18] S.Wang and T.Tsai, "Test Diagnosis of faulty logic blocks in FPGAs", ICCAD, pp. 722-727, 1997.

[19] E.McCluskey, "Verification Testing a pseudoexhaustive Test technique", IEEE tran on computers, vol C-33, No.6, pp. 541-546, june 1984.

[20] H.Michinishi et al, "A test methodology for configurable logic blocks of lookup table based FPGAs", IEICE tran.D-I,vol j79-DI, no.12, pp. 1141-1150, Dec 1996.

[21] W.K.Huang and F.Lombardi,"An Approach to testing Programmable/Configurable Field Programmable Gate Arrays," Proc.IEEE VLSI Test Symp.,pp. 450-455,1996.

[22] C.Jordan and W.P.Marnane, " Incoming Inspection of FPGAs,"Proc.European Test Conf. pp. 371-377, 1993.

[23] XLINIX data book, Field programmable gate arrays,1998.

[24] ALTERA data book, Field programmable gate arrays, 1993.