Test and Diagnosis of Faulty Logic Blocks in FPGAs

Sying-Jyan Wang and Tsi-Ming Tsai
Institute of Computer Science
National Chung-Hsing University
Taichung 402, Taiwan, R.O.C.
E-mail: sjwang@cs.nchu.edu.tw

Abstract

Since Field programmable gate arrays (FPGAs) are reprogrammable, faults in them can be easily tolerated once fault sites are located. In this paper we present a method for the testing and diagnosis of faults in FPGAs. The proposed method imposes no hardware overhead, and requires minimal support from external test equipments. Test time depends only on the number of faults, and is independent of the chip size. With the help of this technique, chips with faults can still be used. As a result, the chip yield can be improved and chip cost is reduced. Experimental results are given to show the feasibility of this method.

1. Introduction

Field programmable gate arrays (FPGAs) can be programmed in the field to implement any logic circuits. They are widely used in rapid system prototyping because of their reprogrammability, and they have also been used in many practical circuits. An FPGA usually consists of configurable logic blocks (CLBs), programmable I/O blocks (IOBs), and programmable interconnects.

Testing faults in general FPGAs has been studied by many researchers [1]-[4]. In these methods the FPGA under test is not mapped to a specific logic function. As a result, usually multiple test sessions are required, with each session dealing with one configuration. Stroud *et al* propose a Built-In Self-Test (BIST) method for FPGAs [5]. This approach is attractive for two reasons. First of all, it requires no extra hardware. Since FPGAs are programmable, the BIST structure is implemented only once during test time. Secondly, the requirement for automatic test equipment (ATE) is much simplified. The ATE only needs to download the configurations that are used during test time, while the actual testing process is conducted by the circuit itself.

Traditional chip-level testing usually deals with fault detection only, while fault diagnosis is often conducted in the system-level. This is because components in chip can hardly be repaired. However, faults in FPGAs can be easily tolerated by not including faulty elements in the final circuit. Therefore, FPGA chips with faults can still be used if we can identify the fault sites. In this paper we propose a chip-level diagnosis methodology for faults in CLBs. Our method is also based on BIST technique, which means the requirement for external ATE support is limited. The amount of testing time is affected by the number of faults on chip only and is independent of the chip size. Since the die yield of larger chips is lower than smaller ones with the same defect density, our method is especially attractive for larger chips.

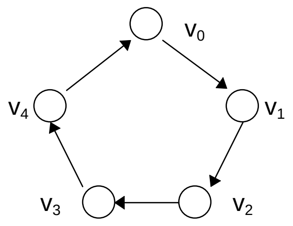
2. Preliminaries

A well-studied model for system-level diagnosis is known as the PMC model [6]. This model is based on the following assumptions. (1) A system is divided into many units, and each unit can test another one. (2) The test outcome is either “pass” or “fail”. (3) If the testing unit is fault-free, the test outcome is always correct. The test outcome obtained by a faulty testing unit is not reliable.

Based on these assumptions, we may model a diagnostic system with a diagnostic graph. A diagnostic graph $G = <V, E, W>$ is a directed graph, where V is the set of vertices, E is the set of edges, and W is the set of weights associated with all edges. A vertex v in V represents a unit in the system. An edge from v_i to v_j in E (denoted as e_{ij}) corresponds to the existence of a test in which unit v_i evaluates v_j. The test outcome a_{ij} associated with e_{ij} may assume the following values:

$a_{ij} =$ 0 if both v_i and v_j are fault-free;
$a_{ij} =$ 1 if v_i is fault-free and v_j is faulty;
$a_{ij} =$ $\times$ if v_i is faulty, no matter what is the status of v_j (i.e., $\times$ is either 0 or 1).

For example, consider the diagnostic graph shown in Fig. 1. Some possible syndromes are shown in Fig. 1. If all units are fault-free, the syndrome would be all 0's. It can be seen that any single fault can be correctly diagnosed since two different faults will not produce the same syndrome. On the other hand, proper diagnosis is not possible if two or more faults are present. For example, the syndrome for fault $\{v_0\}$ and $\{v_0, v_1\}$ may be identical. This system is thus called 1-fault diagnosable since any single fault is guaranteed to be located.



faulty units	a_{01}	a_{12}	a_{23}	a_{34}	a_{40}
None	0	0	0	0	0
v_0	x	0	0	0	1
v_1	1	x	0	0	0
v_0, v_1	x	x	0	0	1
v_0, v_2	x	1	x	0	1

Fig. 1. A diagnostic system

A system of n units is *one-step t-fault diagnosable* if faulty units can be identified without replacement, given that the number of faulty units is at most t [6]. The next theorem provides necessary conditions for such systems.

Theorem 1 [6]: In a one-step t-fault diagnosable system S

1) There must be at least $2t+1$ units.
2) Each unit must be diagnosed by at least t other units.

Many sufficient and necessary conditions have been provided for one-step t-fault diagnosable systems.

Theorem 2 [7]: Let S be a system with the property that no two units test each other. Then S is one-step t-fault diagnosable if and only if each unit is tested by at least t other units.

The above results deal with the diagnosability of a system. Another issue is how to identify the faulty units for a given syndrome. Many diagnosing algorithms have been proposed. An $O(n^{2.5})$ algorithm was given for any diagnosable system [8], and the complexity can be even lower for systems with regular test structure.

3. Test architecture

In order to diagnose faults, first there must be a way to test modules in FPGAs. A candidate for this purpose is BIST [9]. The BIST structure reconfigures part of the functional circuit to be a test pattern generator (TPG) and some other to be an output response analyzer (ORA). The rest circuit consists of the circuit under test (CUT). The TPG is either a linear feedback shift register (LFSR) or a counter. The test inputs are fed to the CUT, while the output responses are collected and analyzed by the ORA. The ORA can be either a signature analyzer or a comparator-based analyzer.

3.1. Basic Architecture

Our test architecture works as follows. An FPGA is divided into disjoint sets of CLBs, where each set can be configured into a TPG and an ORA. Such a set acts as a unit in the PMC model since it is able to test another unit. In Fig. 2 we illustrate the testing method. All the CLBs under test are programmed in the same way; therefore, they perform the same logic function and could be applied with the same test patterns. Thus outputs of the TPG are fed to all CLBs in the set under test, and the results are analyzed by the ORA.

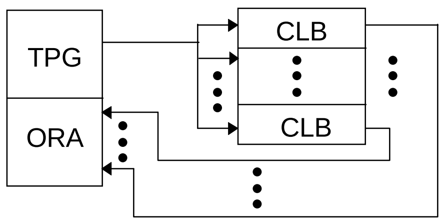


Fig. 2. The connection between a testing unit and a set of CLBs under test

Since each CLB can be programmed in many ways, it is not possible to test any CLB in a single test run. A complete test for a CLB requires several steps, and in each step a CLB is programmed in a particular way.

Definition 1: A *test phase* is the procedure of testing faults in a CLB that is programmed in a particular way.

Definition 2: A *test session* consists of many test phases that completely test CLBs for all modeled faults.

Each test session corresponds to an edge in a diagnostic graph.

Definition 3: A *node* is a set of CLBs that can be reconfigured to a TPG and an ORA when it acts as a testing unit. A node is said to be in *T-mode* when it contains a TPG and an ORA. When the node is corresponding to a tested unit in the PMC model, all CLBs in it are configured in the same way, and the configuration changes from phase to phase. At this moment the node is said to be in *C-mode*.

A node corresponds to a vertex in a diagnostic graph. The only requirement for a node is that it must be large enough to accommodate a TPG and an ORA; and any two nodes are not required to be equal in size. However, if we use homogeneous nodes, the testing and diagnosis procedures can be greatly simplified. Therefore we will assume that all nodes are equal in size throughout the remaining part of this paper.

Definition 4: A *test group* is a set of nodes in which the target number of faulty nodes can be correctly diagnosed.

Our test and diagnosis strategy works as follows. The CLBs on a chip are grouped into nodes, where each node must be able to accommodate at least a TPG and an ORA. Let the maximum number of faulty CLBs allowed on a chip be t. The nodes are again assembled into test groups, where each group must contain at least $2t+1$ nodes. Let the number of nodes in a group be n. We will label these nodes as v_0, v_1, ..., v_{n-1}. Such a group is t-fault diagnosable according to Theorem 2. A node v_i tests v_j, if $j - i = m \pmod{n}$, where $1 \leq m \leq t$. For example, the system in Fig. 1 is a 1-fault diagnosable test group.

After the above procedure having been finished, all faulty nodes can be identified. However, faulty CLBs are not known at this stage since a node contains multiple CLBs. If we need to exactly identify all faulty CLBs, a second step is required. This can be done since we have more than half good nodes and we know the locations of good nodes. The good CLBs are again configured into TPGs and ORAs to locate the faulty CLBs.

3.2. An Efficient Diagnosis Scheme

If an n-unit system is t-fault diagnosable, at least $n \cdot t$ test sessions are required. If the number of faults is smaller than t, then many test sessions are redundant. When the time required to conduct all test sessions is much larger than the time required to carry out the diagnosing algorithm, it is worthwhile to execute the diagnosing algorithm several times so that the required number of test sessions can be reduced.

Consider the test group discussed in Sec. 3.1. This group can be represented by an n-vertex directed graph where an edge (v_i, v_j) exists if $(j-i) \bmod n$ is a member of $\{1, 2, ..., t\}$. We divide the $n \cdot t$ test sessions into t sets, each containing n test sessions. These sets are called T_1, T_2, ..., T_t, where $T_j = \{(v_i, v_{(i+j \bmod n)}) \mid 0 \leq i \leq n-1\}$ for all $1 \leq j \leq t$. An adaptive one-step t-fault diagnosing scheme is possible with the above test sets.

Procedure 1: Adaptive One-Step *t*-Fault Diagnosis

```
PreviousFaults = ∅;
AccumulatedOutCome = ∅
for (i = 1; i <= t; i++) {
    Outcome = Apply_Test_Session(T_i);
    AccumulatedOutcome = AccumulatedOutcome ∪ Outcome;
    Faults = Fault_Diagnosis(AccumulatedOutcome, i);
                /* Diagnosis for i-fault */
    if (Faults == PreviousFaults) break;
    else PreviousFaults = Faults;
}
return faults;
```

Fault_Diagnosis() can be any diagnosis algorithm. In the i-th iteration of the loop, Fault_Diagnosis() tries to solve the given syndrome by assuming that at most i faults exist. We may not be able to reach a consistent result in iteration i since the actual number of faults could be larger than i. If a consistent diagnosis result is not possible at iteration i, the comparison with the previous result is false. The above algorithm says that, if we have the same valid diagnosis results in two successive iterations of the loop, then the remaining test sessions can be omitted since the number of faults in the test group is i.

Theorem 3: Procedure 1 is t-fault diagnosable.

A proof of the correctness of the procedure can be found in [10].

3.3. Analysis

3.3.1. Capability of the Test Architecture. Let a node contain B CLBs, and the number of CLBs on a chip be N. Then the number of nodes on a chip is $\lfloor N/B \rfloor$. If B does not divide N evenly, there will be less than B remaining CLBs that cannot form a complete node. We may ignore these CLBs first. When all correct nodes are identified, these remaining CLBs can be tested separately.

Under the PMC model, at most $t = \lfloor n/2 \rfloor$ faulty nodes can be diagnosed if sufficient tests are applied. Since the interconnects between nodes can be reconfigured in an FPGA, the required tests can always be arranged. Since a chip has $n = \lfloor N/B \rfloor$ nodes, we have $t = \lfloor N/2B \rfloor$. Therefore, a chip with $\lfloor N/2B \rfloor$ or less faulty CLBs is guaranteed to be correctly diagnosed, and in the optimal condition up to $\lfloor N/2 \rfloor \times B$ faulty CLBs can be diagnosed.

3.3.2. Structure of a Test Block. The constraint for a node is that it must be able to accommodate a TPG and an ORA. The proposed model does not impose an upper bound on the number of CLBs in a node. However, we want to make a node as small as possible. As shown in Fig. 2, the output of a TPG drives all CLBs in a C-mode node (i.e., the fanout of a TPG output line is B). Therefore, a large B means a large load for the TPG. If B is larger than the driving capability of a TPG, output buffers must be provided to the TPG. The second reason for a smaller B is the diagnostic capability. Since the maximum number of faulty CLBs that are guaranteed to be diagnosed is $\lfloor N/2B \rfloor$, smaller nodes imply more faulty CLBs are diagnosable. For these reasons, we assume that a node can accommodate exactly a TPG and an ORA.

Let the sizes of a TPG and an ORA be B_{TPG} and B_{ORA}, respectively. By the above arguments,

$$B = B_{TPG} + B_{ORA} \tag{1}$$

Define the following parameters of a CLB:

O: Number of outputs of a CLB
I: Number of inputs of a CLB
F: Number of flip-flops in a CLB

In an exhaustive test of a CLB for a given configuration, all 2^I input vectors are required. This means that the TPG must have I independent output signals, which implies I flip-flops are required:

$$B_{TPG} \geq \lceil I/F \rceil \qquad (2)$$

In practice, if two input lines do not affect the same output, they may be applied with the same signal to achieve pseudoexhaustive testing [9]. In this case, p ($< I$) signals may be enough, and some of the p lines may be connected to two or more inputs of a CLB:

$$B_{TPG} \geq \lceil p/F \rceil \qquad (2a)$$

Since the outputs of all CLBs in a node under test are analyzed by the same ORA, we have:

$$B \times O \leq B_{ORA} \times I \qquad (3)$$

3.3.3. A Test Group. If we want to achieve maximum diagnosability, only one test group is allowed on chip. In this case $\lfloor N/2B \rfloor$ faulty nodes can be identified. On the other hand, if it is known that the number of faulty nodes on a chip, t, is much less than $\lfloor N/2B \rfloor$, it is possible to have smaller test groups. The only requirement for a test group is that it must contain at least $2t$+1 nodes.

Employing multiple test groups does not improve diagnosability. The advantage of smaller test groups is the reduced diagnosing time. Since all test groups conduct their test sessions simultaneously and the time required for diagnosis is $O(n^{2.5})$, the diagnosing time is greatly reduced for smaller test groups.

4. Case Study: XC4000 FPGA FAMILY

We implemented the above test architecture with Xilinx XC400 FPGAs, and the details are discussed here.

4.1. Testing a CLB

A simplified diagram of a CLB in Xilinx XC4000 Family is shown in Fig. 3. This CLB has 13 inputs (I = 13), in which one is clock input (K), nine signals are input to the combinational part (F1 to F4, G1 to G4, and H1), and the other three are for the sequential part (DIN, S/R, and EC). There are four outputs in a CLB (O = 4). The combinational part consists of 3 Look-Up Tables (LUTs) and three multiplexers (MUXs) whose outputs are H1, X, and Y, respectively. The sequential part contains the remaining components: two D flip-flops (F = 2), the S/R control, and the remaining MUXs.

The LUTs are made of SRAM. To test the LUTs, each bit in them has to be set to both 0 and 1. Therefore, at least two phases are required to exercise all possible faults in the LUTs. There are 4-to-1 MUXs in a CLB. As a result, we need at least four test phases so that each input-to-output connection of these MUXs can be exercised. We found that four test phases are enough to exercise all possible configurations in the CLB.

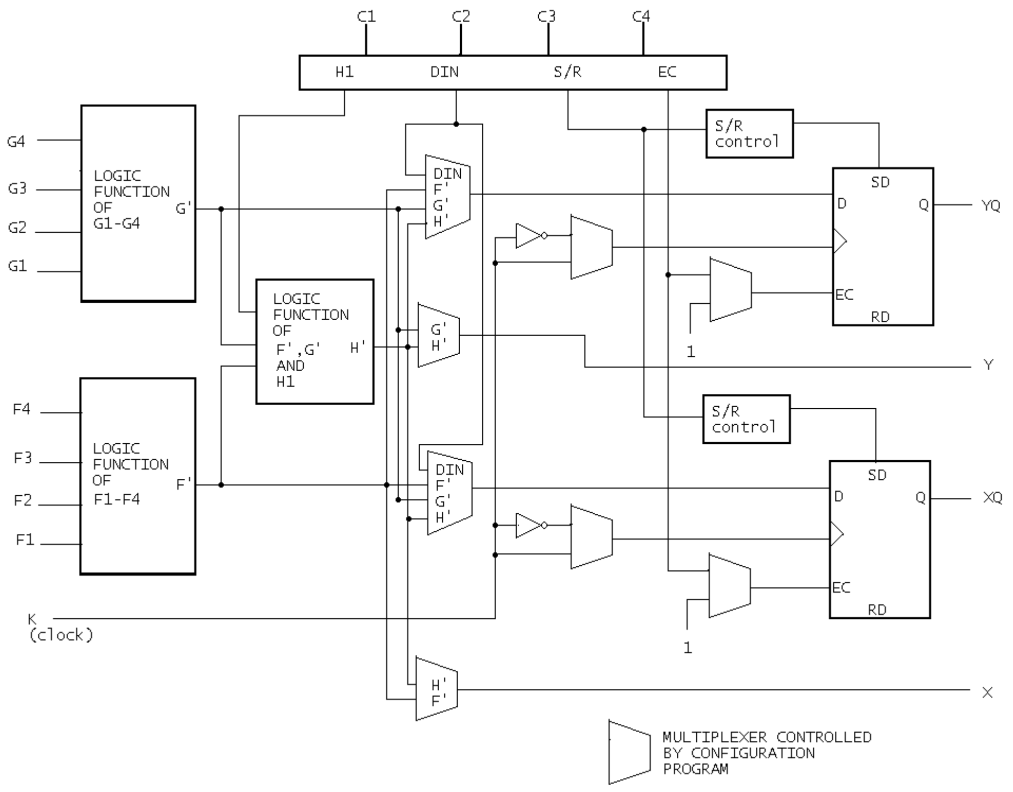


Fig. 3. A simplified diagram of a CLB in XC4000

- **Phase 1:** The LUTs are configured as Exclusive-OR of the nine inputs. F′ is connected to both flip-flops, G′ is connected to Y, and H′ is connected to X.
- **Phase 2:** The LUTs are configured as Exclusive-NOR of the nine inputs. G′ is connected to both flip-flops, F′ is connected to X, and H′s is connected to Y.
- **Phase 3:** The LUTs implement XOR, and DIN is connected to both flip-flops.
- **Phase 4:** The LUTs implement XNOR, and H′ is connected to both flip-flops.

The combinational part is tested in the first two phases. Flip-flops are tested in all four phases. In order to fully test the upper MUXs, in Phase 1 the connections are (C1, C2, C3, C4) $\Rightarrow$ (H1, DIN, S/R, EC) (i.e., H1 is connected to C1, etc.). In Phase 2 the connection is (C2, C3, C4, C1) $\Rightarrow$ (H1, DIN, S/R, EC), etc. As a result, all connections are exercised in four phases.

In each phase, we apply pseudo-exhaustive test vectors to test all faults in a CLB. The question is how many independent signals are required to carry out the test. In the 13 inputs to the CLB, the clock (K) is fed by the system clock and thus is not considered. The three input signals (DIN, S/R, and EC) for the sequential part are independent of the combinational inputs (that is, they do not affect the same output), so these three signals can be connected to any 3 of the nine combinational inputs. Therefore, we need no more than 9 test signals. A further reduction can be made. If we fed all 2^8 vectors to inputs F1 to F4 and G1 to G4, both LUTs F′ and G′ are exhaustively tested, and all four combinations (00, 01, 10, 11) will appear on F′ and G′ since they are either XOR or XNOR. Now in order to fully test LUT H′, we need $2^3 = 8$ independent inputs for each configuration. The third input (H1) can be connected to any one of the other eight combinational inputs to get exhaustive input vectors for H′. Therefore, we need only 8 independent signals to test each configuration of the CLB exhaustively (i.e., $p = 8$ in Eq. 2a).

4.2. Test Blocks

In our design, the TPG is made of a type I LFSR [9] which implements the polynomial $x^8+x^6+x^5+x+1$. This LFSR generates 255 patterns (2^8-1) except the all 0 vector. In order to get all 256 exhaustive vectors, we reset all FFs in the first cycle, and then set the first bit in the second cycle. The rest vectors will be automatically generated. This TPG consists of 4 CLBs ($B_{TPG} = 4$).

We use the comparator-based ORAs because they are easier to be implemented and do not suffer the alias problem. An ORA compares the outputs of all tested CLBs; whenever there are inconsistent results, an error bit is set to record the fault. The ORA design also consists of 4 CLBs ($B_{ORA} = 4$). Therefore, a test node contains 8 CLBs ($B = 8$). It is easy to verify that this design satisfies conditions in Sec. 3.3.2.

4.3. Experimental Results

We experiment our method on an XC40003A PC84 FPGA chip. An XC4003A chip has 100 CLBs arranged in a 10×10 array. Since $B = 8$, this chip may have up to $\lfloor 100/8 \rfloor = 12$ nodes. However, it is difficult to construct nodes from CLBs locating at the chip boundary. Therefore, we construct only 10 nodes in this case. The remaining 20 CLBs are considered after we have identified faulty nodes. Fig. 4 shows the structure of our design on an XC3003A chip. All nodes locate at center part of the chip, while the top and bottom rows of CLBs are not configured.

Since CLBs are configurable, we can “inject” faults into CLBs by simulating the effects of logic faults. We consider two types of faults: stuck-at faults and open faults. We simulate the effect of a stuck-at fault by connecting the faulty line to logic “0” or “1”. To simulate the effect of an open fault, all we have to do is to disconnect the faulty line from its driving source. Unfortunately, we are unable to simulate the bridging fault. The reason is that, if we try to short two lines that are driven by different sources, a DRC error occurs and the design is not realizable. In our experiment we inject 48 faults into the chip, in which there are 25 stuck-at faults and 23 open faults. These faults appear in 17 different CLBs. Eleven faulty CLBs locate in four test nodes, and the remaining 6 faulty CLBs are in the top and bottom rows. Fig. 4 shows the locations of faulty CLBs and faulty nodes.

It requires three different stages to identify all faulty CLBs, and the three stages are described as follows.

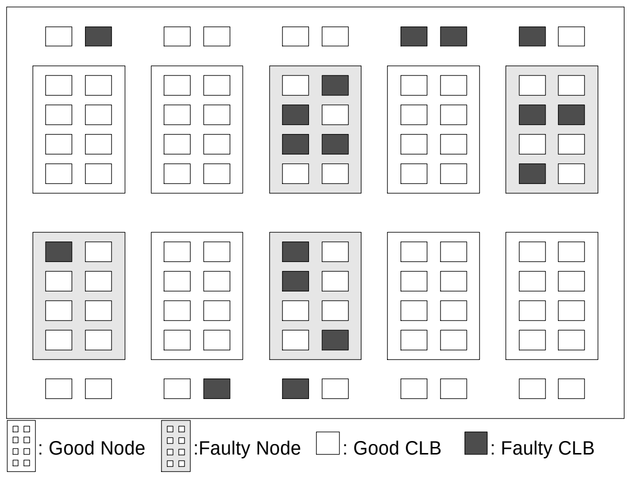


Fig. 4. Experimental environment

Stage 1: Identify faulty nodes. Since there are 10 nodes on the chip, we are able to identify all four faulty nodes. We apply test sets T_1, T_2, T_3, and T_4 in sequence. Each test set consists of 10 test sessions. These test sessions, however, cannot be completed in one step for several reasons. First of all, each node is in C-mode in one session while it is in T-mode in another session. Since a node cannot be in both C-mode and T-mode, it requires at least two steps to execute a test set. Secondly, due to the limited routing resource, it may not be possible to have maximum test sessions running concurrently.

Stage 2: Identify faulty CLBs in faulty nodes. The locations of faulty CLBs are known at this stage. One good node is configured as a TPG. Output of the TPG drives two nodes: one is faulty and the other is faulty-free. The two nodes are programmed in the same way during test time. A comparator compares the output responses of the two tested nodes. The TPG design is the same as before, while the comparator is somewhat different here.

Stage 3: Identify faulty CLBs in the remaining part. The remaining CLBs are divided into three nodes. We repeat the process in stage 2 to identify faulty CLBs. It requires 3 test sessions for the remaining CLBs.

The time used to apply all test sessions in Stage 1 is:

$$\text{Time}_{\text{Stage\_1}} = \sum_{i=1}^{t}(D + C \times 2^{v}) \times S_i \times P \qquad (4)$$

in which D is the time for download, C is the clock cycle time, v the number of outputs of a TPG, S_i is the number of steps required to apply test sessions in set T_i, and P is the number of phases in a session. Among them, D, C, v, and P are constant for a given FPGA technology.

The test time of our method depends on the number of diagnosable faults t and the number of steps to arrange all the test sessions (form Eq. 4). It is independent of the chip size. Therefore, we need to reduce the summation of all S_i if we want to reduce the test time. Procedure 1 in Sec 3.2 is developed for this purpose.

The time required to complete Stage 2 is:

$$\text{Time}_{\text{Stage\_2}} = (D + C \times 2^{v}) \times P \times t_F \qquad (5)$$

where t_F is number of actual faulty nodes. The time required to complete Stage 3 is:

$$\text{Time}_{\text{Stage\_3}} = (D + C \times 2^{v}) \times P \times R \qquad (6)$$

where R is the number of remaining nodes.

The following parameters are valid for the XC4000 family CLBs. A test session requires four phases (P = 4). In each phase we need to download the configuration once and apply 2^8 test vectors (v = 8). Our experimental results show that downloading a bitstream needs about 7.8 seconds. We use a 8MHz clock, that is, C=1.25×10^{-7} sec. The time required to apply all test vectors in a test phase is thus 2^8×1.25×10^{-7} = 32μs. Since the test application time is negligent compared with the time for downloading, we can ignore this term.

We need 15 steps in the first stage. For the second and third stage, we have t_F = 4 and R = 3. Put everything together, we need 7.8×4×(15+4+3) = 686.4 seconds.

In terms of diagnosability, all 17 faulty CLBs in the experiment are properly located, as we expect.

5. Concluding Remarks

In this paper we present a method to diagnose faulty CLBs in FPGAs. The advantages of this method include: (1) no hardware overhead, (2) all tests and responses are processed on chip, (3) test time depends on the number of faults only, and is independent of the chip size. With this method, FPGAs with faulty CLBs can still be used.

A significant part of the FPGA die area is dedicated to the interconnections. Faults on interconnections are not considered in this paper; we will explore this issue in the future.

REFERENCES

[1] T. Liu, W.K. Huang, and F. Lombardi, "Testing of uncustomized segmented channel FPGAs," in *Proc. ACM Intl. Symp. on FPGAs*, pp. 125-131, 1995.

[2] T. Inoue, H. Fujiwara, H. Michinishi, T. Yokohira, and T. Okamoto, "Universal test complexity of field-programmable gate arrays," in *Proc. 4th Asian Test Symp.*, pp. 259-265, 1995.

[3] W.K. Huang and F. Lombardi, "An approach for testing programmable/configurable FPGAs," in *Proc. IEEE 14th VLSI Test Symp.*, pp. 125-131, 1995.

[4] H. Michinishi, T. Yokohira, T. Okamoto, T. Inoue, and H. Fujiwara, "A test methodology for interconnect structures of LUT-based FPGAs," in *Proc. 5th Asian Test Symp.*, pp. 68-74, 1996.

[5] C. Stroud, S. Konala, P. Chen, M. Abramovici, "Built-in self-test of logic blocks in FPGAs," in *Proc. IEEE 14th VLSI Test Symp.*, pp. 387-392, 1996.

[6] F.P. Preparata, G. Metze, and R.T. Chien, "On the connection assignment problem of diagnosable systems," *IEEE Trans. Electronic Comput.*, EC-16, pp. 848-854, Dec. 1967.

[7] S.L. Hakimi and A.T. Amin, "Characterization of the connection assignment of diagnosable systems," *IEEE Trans. Comput.*, C-23, pp. 86-88, 1974.

[8] A.T. Dahbura and G.M. Masson, "An O($n^{2.5}$) fault identification algorithm for diagnosable systems," *IEEE Trans. Comput.*, C-33, pp. 486-492, June 1984.

[9] M. Abramovici, M.A. Breuer, and A.D. Friedman, *Digital Systems Testing and Testable Design*, W. H. Freeman and Company, 1990.

[10] S.-J. Wang, "Adaptive system-level diagnosis and its application," in *Proc. 1997 Pacific Rim Intl. Symp. On Fault-Tolerant Systems*.