VLSI Design, 2002 Vol. 15 (1), pp. 397–406

Taylor & Francis
Taylor & Francis Group

Fault Detection and Fault Diagnosis Techniques for Lookup Table FPGAs

SHYUE-KUNG LU[a,*], FU-MIN YEH[b,†] and JEN-SHENG SHIH[a]

[a]*Department of Electronic Engineering, Fu Jen Catholic University, Taipei, Taiwan, ROC;* [b]*Electronics Systems Division, Chung-Shan Institute of Science and Technology, Taipei, Taiwan, ROC*

(Received 28 February 2001; Revised 23 May 2001)

In this paper, we present a novel fault detection and fault diagnosis technique for Field Programmable Gate Arrays (FPGAs). The cell is configured to implement a *bijective* function to simplify the testing of the whole cell array. The whole chip is partitioned into disjoint one-dimensional arrays of cells. For the lookup table (LUT), a fault may occur at the memory matrix, decoder, input or output lines. The input patterns can be easily generated with a k-bit binary counter, where k denotes the number of input lines of a configurable logic block (CLB). Theoretical proofs show that the resulting fault coverage is 100%. According to the characteristics of the bijective cell function, a novel built-in self-test structure is also proposed. Our BIST approaches have the advantages of requiring less hardware resources for test pattern generation and output response analysis. To locate a faulty CLB, two diagnosis sessions are required. However, the maximum number of configurations is $k + 4$ for diagnosing a faulty CLB. The diagnosis complexity of our approach is also analyzed. Our results show that the time complexity is independent of the array size of the FPGA. In other words, we can make the FPGA array *C-diagnosable*.

Keywords: FPGA; Testing; Diagnosis; C-diagnosable; Bijective; CLB

INTRODUCTION

Field Programmable Gate Arrays (FPGAs) are regularly constructed with configurable logic blocks (CLBs) and input/output blocks (IOBs) communicated with interconnects and switches. They are a popular type of component for emulation and rapid prototyping of complex digital systems. Their applications include microprocessors and telecommunication chips [1]. An FPGA can be configured to implement combinational or sequential logic functions. Because of its low manufacturing cost, short turnaround time and field programmability, it has been widely used in many applications [1]. Several FPGA architectures have been developed for different applications. The most widely used type is the *lookup table* (LUT) FPGA, in which the functional unit consists of several LUTs. This type of FPGA can be reprogrammed any number of times. In this paper, we will focus on the testing of LUT-based FPGAs.

In recent years, VLSI technology keeps greatly increasing the degree of circuit integration and the rapid development in packaging technology greatly reduces the controllability and observability of internal nodes. This significantly complicates testing of the system. If chips are not fully tested after being manufactured, they become low-quality products. Therefore, manufacturers of FPGAs also face the same problem: seek for an efficient test methodology to ensure the quality of their products. The testing of FPGAs falls into two categories: testing of unprogrammed FPGAs (*configuration-independent* testing) and testing of programmed FPGAs (*configuration-dependent* testing). In configuration-independent testing, no assumptions are made about the function that the FPGA will be configured by the user. On the other hand, configuration-dependent testing involves testing whether a configured FPGA is fault free or not. Some techniques for testing FPGAs can be found in Refs. [2,3]. Here, we focus on the testing of unprogrammed FPGAs, for which many research results have been proposed [4–18].

In Ref. [4], an array-based technique is proposed to test LUT-based FPGAs. The number of chip configurations to fully test all CLBs is the same as to test a single CLB with perfect controllability and observability. Universal and C-diagnosable techniques are introduced in Refs. [5,6].

*Corresponding author. E-mail: sklu@ee.fju.edu.tw
†E-mail: fmyeh@tpts5.seed.net.tw

ISSN 1065-514X print/ISSN 1563-5171 online
DOI: 10.1080/1065514021000012011

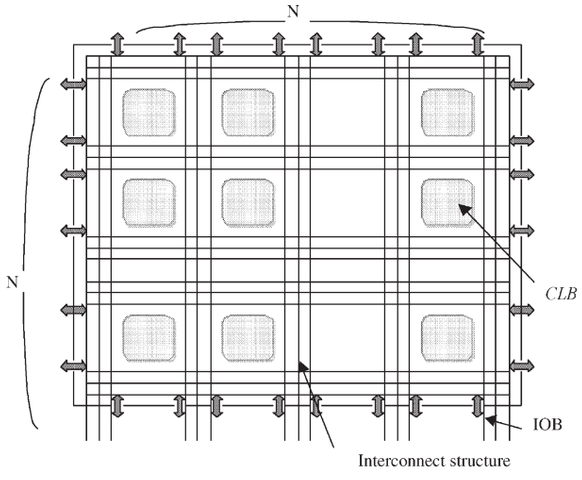

FIGURE 1 FPGA architecture.

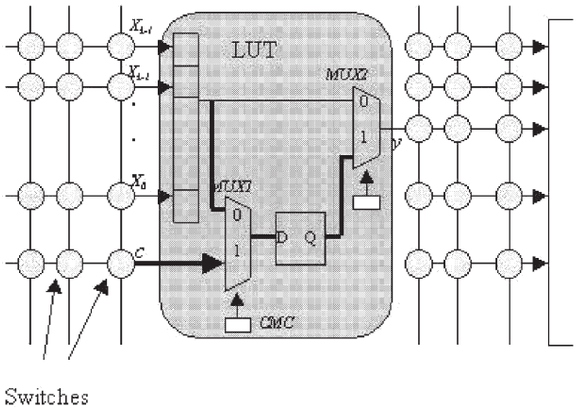

FIGURE 2 The structure of a CLB.

C-diagnosability denotes that the numbers of test patterns (TPs) and test configurations (TCs) are both constant regardless of the size of an FPGA. The proposed fault diagnosis approaches are universal—regardless of the logic functions to be realized and the time required to diagnose whether an FPGA is independent of the array size. The methodologies proposed in Refs. [7,8] connect each row of an FPGA as a one-dimensional unilateral array for single stuck-at faults. In Ref. [9], BIST structures for FPGAs are proposed. The FPGA is repeatedly configured as a group of C-testable *iterative logic arrays* (ILAs). Test generation and response analysis are performed using internal BIST circuits. In Refs. [11,12], test approaches for interconnects of FPGAs are proposed. Moreover, the testing of multiplexer-based FPGAs can be found in Ref. [19].

An FPGA is structured as a $N \times N$ array of CLBs, it can be naturally viewed as a two-dimensional array or N one-dimensional arrays as shown in Fig. 1. A general structure as shown in Fig. 2 for the basic CLBs is assumed. This structure can be found in various Xilinx FPGA families. Unfortunately, a CLB does not have enough outputs to drive the inputs of a neighboring CLB. This problem makes it difficult to configure an FPGA as an ILA. Therefore, if we treat an FPGA as an $N \times N$ array of CLBs directly, then we cannot exploit the testing advantages of ILAs. Fortunately, this problem have been solved in Ref. [9], which uses some CLBs as *helpers*—auxiliary cells whose goals are to generate "locally" the missing inputs and to provide separate propagation paths for the output responses of blocks under test (BUTs). However, although some portion of the helper CLB may also be tested alongside the BUT, it must be fully tested in a different test session. Therefore, more chip configurations and TPs are required to complete the test process.

In order to deal with this problem, a novel cell structure is proposed here. In this paper, a *cell* is a CLB with k input lines and k output lines, respectively, where k denotes the number of inputs of an LUT. Then, the whole chip is partitioned into N disjoint one-dimensional arrays of cells. We assume that in each linear cell array, there is at most one faulty cell. In other words, multiple faulty cells can be detected if they exist in different linear arrays. By using this cell structure, the test pattern generator (TG) and the output response analyzer are easy to implement. All the cells in the FPGA can be tested simultaneously to speed up the test process. Moreover, since each cell is locally connected to its neighboring cells, there requires no considerable demands on the global routing resources. This increases the probability of successful routing [9]. During testing, the configurations of each cell will make its function *bijective*, which is helpful for applying pseudoexhaustive TPs to each cell and propagating errors to observable outputs. In order to detect all the faults modeled, $k+2$ configurations are required. For each configuration, a *minimal complete input sequence* (consisting of all input combinations for a cell) is applied to the leftmost cells of each linear array from IOBs and the outputs of the rightmost cells can be observed directly from the remaining IOBs.

In order to simplify our discussion, some definitions are given first.

DEFINITION An *unprogrammed cell* is a CLB, which can implement 2^k functions. The output of the CLB is $y = x_i$ for test configuration TC_i, $0 \le i \le k-1$.

DEFINITION A *programmed cell* with function f is a combinational machine (Σ,Δ,f), where $f : \Sigma \rightarrow \Delta$ is the programmed cell function, and $\Sigma = \Delta = \{0,1\}^k$, k denotes the number of inputs of a CLB.

DEFINITION A *complete* or *exhaustive* input sequence σ for a programmed cell with function f is an input sequence consisting of all possible input combinations for the cell. A *complete output sequence* is defined analogously.

DEFINITION A *minimal complete input sequence* θ for a programmed cell is a shortest complete input sequence (which has a length of 2^k, i.e., $\theta = \theta_1, \theta_2, \ldots, \theta_n$).

DEFINITION We say that the function f of a programmed cell is *bijective* when $\forall \theta_1 \neq \theta_2$, $f(\theta_1) \neq f(\theta_2)$, and the length of the minimal complete input sequence and that of the minimal complete output sequence are identical, i.e., $|\Sigma| = |\Delta|$.

According to the bijective characteristics of the cell function, a novel built-in self-test structure is also proposed in this paper. The input patterns can be easily generated with a k-bit binary counter and the output response are analyzed with a checksum analyzer or comparator. Our BIST approaches have the advantage of requiring fewer hardware resources for TP generation and output response analysis. The number of configurations for our BIST structures is $2k + 4$. To locate a faulty CLB, two test sessions are required. However, the maximum number of configurations for diagnosing a faulty CLB is $k + 4$. The complexity for fault diagnosis is also derived. Our results show that the time complexity is independent of the array size of the FPGA. In other words, we can make the FPGA array *C-diagnosable* with our approach. Our paper is organized as follows. Second section presents the preliminaries for testing FPGAs. Third section depicts the fault detection approach. Built-in self-test structures are shown in the fourth section. Fault Diagnosis technique is presented in the fifth section. Sixth section analyzes the diagnosis complexity and comparisons with other approaches are given. Finally, seventh section concludes this paper.

PRELIMINARIES

The structure of a CLB is shown in Fig. 2, which consists of one LUT, two multiplexers (MUX1 and MUX2), and one D flip–flop (DFF). Let k denote the number of input lines of an LUT, then an LUT can implement 2^n logic functions, $n = 2^k$. The configuration memory cells (CMCs) are used to configure its logic functions. When programming an FPGA, we load the bit patterns corresponding to the function's truth table into the CMCs and the interconnection networks are also configured. Such a programming process is called a *configuration*. It is well known that an unprogrammed FPGA can be configured to implement different logic functions. Therefore, testing this category of FPGAs implies testing of all possible functions, which can be realized. The procedure for testing CLBs is by repeatedly implementing a *TC* and alternately applying *TPs* to this configuration. Since an LUT can realize 2^n different functions, it is impractical to test each function exhaustively.

Fortunately, with our approach, the number of TPs required for each TC is the same—a *minimal complete input sequence* (defined later) of a cell. The cell array is pseudoexhaustively tested by applying exhaustive tests to each cell. Our TPs are independent of the size of the FPGA, i.e., they scale well with increasing FPGA size. Therefore, the remaining problem is to determine how to configure the cells in order to detect all the faults modeled using the defined fault models. It is evident that TC is very time-consuming. In general, the time for TC is much greater than applying of TPs. Therefore, to test an FPGA, we must seek out an efficient approach, which requires less number of TCs and covers all the programmable resources of FPGAs. Taken literally, the proposed approach must speedup the test process and guarantee the quality of the products under test. It should also be noted that the proposed approach must not place considerable demands on global routing resources within the FPGA and increase the probability of successful routing.

The fault model used in this paper is described in the following, and has been shown to be suitable for FPGAs [2,5,6].

1. For an LUT, a fault may occur at the memory matrix, decoder, inputs and outputs of a CLB. A faulty memory matrix has some memory cells that are incapable of storing the correct logic values (stuck-at 1 or stuck-at 0 may occur at a memory cell). If a fault occurs at the decoder, then incorrect access, non-access and multiple access faults may occur. For the input and output lines, stuck-at faults are considered.
2. A multiplexer is a group of switches controlled by the configuration memory bits. Only one switch is allowed to be on. Either none or more than one on-switches is invalid. If all the switches are off, we assume the output is either stuck-at-1 or stuck-at-0. Alternatively, if two switches are on simultaneously, then the output is the logic wired-OR of the selected inputs. In other words, switch stuck-on and switch stuck-off fault models [19] are adopted. For a DFF, we adopt the functional fault model. A fault may cause the flip–flop to be unable to receive data, to be triggered by the correct clock edge, or to be incapable of being set or reset [8].

We assume that at most one faulty cell exists in each linear cell array. Otherwise, fault masking might occur. However, multiple faulty cells existing in different linear arrays can also be tested since all linear arrays are tested independently.

In general, the number of inputs, k, of a CLB is larger than its output counts. As described in previous sections, it is not beneficial to view an FPGA as an $N \times N$ array of CLBs. This is since a CLB does not have enough outputs to drive the inputs of a neighboring CLB and it is impossible to construct an iterative logic array directly. If we treat an FPGA as an $N \times N$ array of CLBs, we cannot exploit the testing advantages of ILAs. Therefore, many test approaches for ILAs cannot be applied for FPGA testing. To solve this problem, we define a *cell* as a CLB with k input lines and k output lines as shown in Fig. 3. In this figure, $k - 1$ input lines of a CLB are forwarded to the outputs of the cell directly. The output of the CLB is $y = x_i$ for test configuration i, $1 \leq i \leq k$. Therefore,

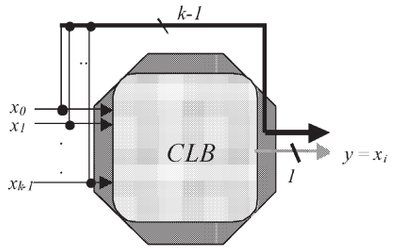

FIGURE 3 Architecture of a cell.

we can regroup an FPGA consisting of an $N \times N$ array of CLBs into an $N \times N$ array of cells. An FPGA consists of a CLB array as shown in Fig. 4, where CLB_{ij} denotes the CLB in row i and column j, $0 \leqq i, j \leqq N-1$. The corresponding cell array of Fig. 4 is shown in Fig. 5. Thereafter, the outputs of one cell can be used to drive its neighboring cells. Therefore, N disjoint linear arrays of cells are obtained and their inter-cell communications are local (between adjacent cells). This will not violate the routing limitations of FPGAs and increase the probability of successful routing. One of the main advantages of our approach is that all cells are tested simultaneously in the same test session. However, the helpers used in Ref. [9] must be tested in a different session. Therefore, our approach will result in less test configurations and increase the test speed significantly.

FAULT DETECTION FOR FPGAS

In this section, new conditions for C-testability of programmable/configurable arrays are proposed. Since our test approach uses a cell as the basic test element instead of using a CLB, all the cells in an FPGA can be tested concurrently as described in the following section.

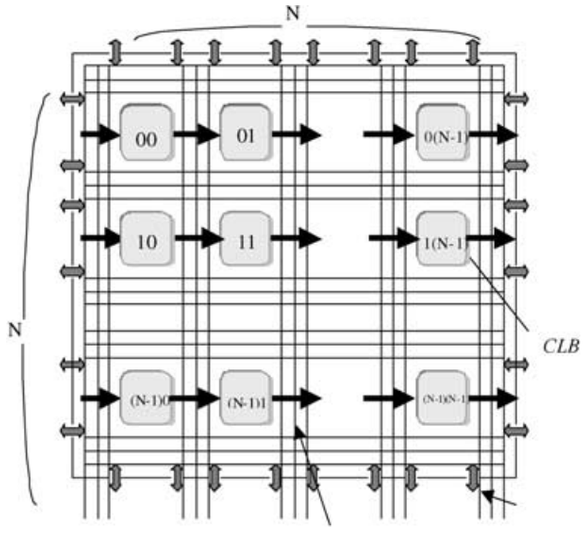

FIGURE 4 A $N \times N$ array of CLBs.

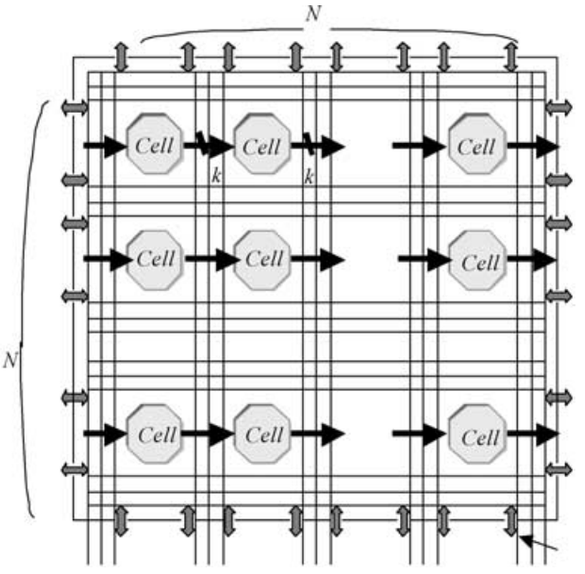

FIGURE 5 The corresponding cell array of Fig. 4.

Testing of LUTs

We assume initially that a CLB consists of a single LUT. Then all the input lines of the leftmost cells and the output lines of the rightmost cells for each linear array are connected to IOBs, so that they can receive their TPs and observe the corresponding outputs, respectively. The number of IOBs required for inputting TPs and observing the results is at most $2kN$. However, if all linear arrays use the same IOBs to input TPs and the BIST structure is used, the number of IOBs used will be reduced significantly. This number will not tend to exhaust the limited I/O resources available at the peripheral of the chip. During testing, the k output lines of cell_{pq} are appropriately connected to the k input lines of $\text{cell}_{p(q+1)}$ $\leqq$ $p \leqq N-1$ and $0 \leqq q \leqq N-1$. This connection structure is shown in Fig. 6, where each array receives its TPs from TG through the same IOBs. The outputs are then sent to the output response analyzer so that the responses can be analyzed in order to produce a pass/fail indication.

After the interconnection network has been configured, there are still three questions to be answered before the LUTs can be tested. (1) Which test set should we apply to test all the linear arrays and ensure acceptable fault coverage? (2) Verifying a cell function involves generating inputs for each cell (i.e., controlling the cell), and propagating faults from the cell (i.e., observing the cell). How can we solve the controllability and observability problems for FPGAs? (3) Unlike general iterative logic array structures, each cell in an FPGA can be configured to have up to 2^n functions. What functions should we choose to ensure their fault detection capability and minimize the number of configurations? The first question can be answered by sending pseudoexhaustive TPs to each cell. The answers to the last two questions can be found in the following paragraphs.

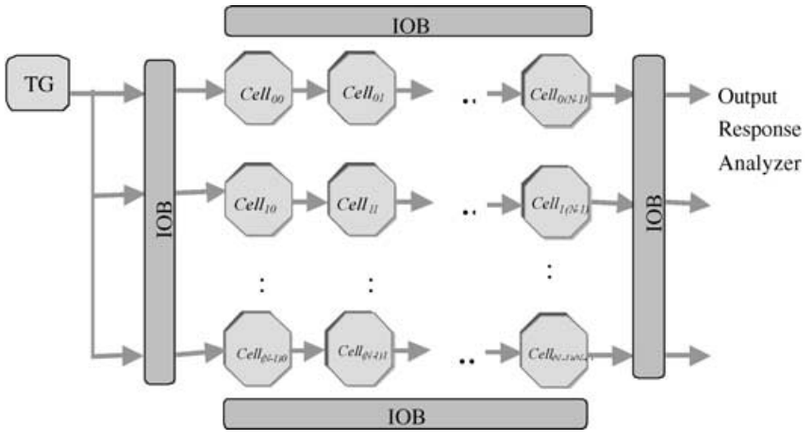

FIGURE 6 The connection structure during testing.

Since each cell has k input lines and output lines, respectively, the function of a cell can be described with a truth table. We also assume that the truth tables are fully expanded, i.e., there are no don't-care terms. In reality, this is always true. The LUT input is assumed to be represented by the k-bit word, $x_{k-1}, x_{k-2}, \ldots, x_0$, while the cell output is represented by the word, $y_{k-1}, y_k, \ldots, y_0$. Let $\alpha \equiv \alpha_{k-1}, \alpha_{k-2}, \ldots, \alpha_0$ denote the output assignments of the corresponding output variables, where α_i denotes the assignment of variable y_i. Now we make the test configuration TC_0 for each cell such that the cell function, f^0, has the assignment α^0. Since for test configuration TC_0, the output of the CLB equal to x_0, then $y_i = x_i$, $0 \leqq i \leqq k-1$. In other words, the outputs of the cell are equal to the inputs. Therefore, the outputs of the cell contain all possible binary k-bit values. In other words, the cell function is configured to be *bijective*. The assignment α^k for configuration TC_k is the complement of assignment α^0. An example of a test configuration TC_0 and TC_k with $k = 3$ is shown in Fig. 7.

X_2	X_1	X_0	Y_2	Y_1	Y_0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	1	0	0
0	1	1	1	0	1
1	0	0	0	1	0
1	0	1	0	1	1
1	1	0	1	1	0
1	1	1	1	1	1

(a) TC_0

X_2	X_1	X_0	Y_2	Y_1	Y_0
0	0	0	1	1	1
0	0	1	1	1	0
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	0	0	0

(b) TC_k

FIGURE 7 Test configuration TC_0 and TC_k.

OBSERVATION If TC_0 is a bijective configuration, then TC_k is also a bijective configuration.

Proof This follows directly from the definition of a bijective cell function. □

Without the loss of generality, we can reconfigure the LUT such that $y = x_i$ for TC_i and all the other input variables are directed as the output variables. Therefore, the output variable assignments α^i can obtain the required TPs to completely test a cell.

OBSERVATION For a fault-free cell, if a minimal complete input sequence δ is applied to the inputs of a cell with configuration TC_i, $0 \leqq i \leqq k$, then the output sequence will also be a complete sequence if it is fault-free.

Proof Since the cell function is also bijective with configuration TC_i, $0 \leqq i \leqq k$, then if the cell is fault free, the outputs will also be a complete sequence since the outputs are the permutation of the inputs. □

THEOREM 1 If a minimal complete input sequence δ is applied to the inputs of a cell with test configuration TC_k and TC_0, respectively, then all stuck-at faults in the LUT memory cells can be detected with *100%* fault coverage.

Proof Assume that a stuck-at-0 (1) fault occurs at the ith memory cell in LUT of a cell, then the ith entry of y will have the same value for configurations TC_0 and TC_k. However, since TC_k and TC_0 make the cell function bijective, their corresponding assignments are complemented. This fault will make either α^0 or α^k to be not bijective. It is evident that this fault can be detected by examining whether the output is a complete output sequence. Therefore, we conclude that all stuck-at faults in the LUT can be detected with 100% fault coverage. □

THEOREM 2 If a minimal complete input sequence δ is applied to the inputs of a cell with configurations

$TC_{k-1}, TC_{k-2}, \ldots$, and TC_0, respectively. Then any fault within the decoder can be detected.

Proof If a fault occurs within the decoder, then incorrect access, non-access or multiple access faults may occur. All these faults must be detected by δ. The proofs for each type of faults are as follows. □

Incorrect access faults If the decoder cannot generate the correct memory address, then a faulty memory address is accessed. That is, if we try to access the *i*th memory cell of an LUT, then the *j*th memory cell will be accessed if there exists a incorrect access fault, $i \neq j, 0 \leq j \leq 2^k - 1$. A large number of such faults may exist in a cell. To test this type of faults, we must first activate this fault. In other words, we must seek for a configuration such that the stored values in the *i*th and *j*th memory cells are different, i.e. $MC(i) \neq MC(j)$. Fortunately, with the configurations $TC_{k-1}, TC_{k-2}, \ldots, TC_0$, there exists at least one such configuration that the accessed values of the correct and faulty memory addresses will have different stored values. If the fault is activated by a specific configuration, then its corresponding outputs will not again be a complete output sequence. This is helpful to detect this fault. Therefore, we conclude that all the incorrect access faults can be detected with 100% fault coverage.

Non-access faults If a non-access fault exist, then the output of the LUT will stick at 0 (1). Then we can treat it as a stuck-at 0(1) fault. This type of faults can be detected with configurations TC_k and TC_0 as described in previous paragraph. Therefore, we conclude that non-access faults can be detected with 100% fault coverage.

Multiple access faults. If a multiple access fault occurs within a CLB, then the output value of this CLB is equal to the wired-AND (wired-OR) of the stored values of all the accessed addresses. That is, the output value $y = MC(i) + MC(j)$ (wired-AND) or $y = MC(i) \cdot MC(j)$, where $MC(i)$ and $MC(j)$ denote the values of the correct and extra accessed memory cells, $i \neq j$ and $0 \leq j \leq 2^k - 1$. To activate this type of faults, there must exist a configuration such that $MC(i) = 0$ and $MC(j) = 1$ (wired-*OR*) or $MC(i) = 1$ and $MC(j) = 0$ (wired-*AND*). Then the wired-AND (wired-OR) output of this CLB will has logic value 0/1 (wired-AND/wired-OR) and the faulty effect is activated. Fortunately, the configurations TC_{k-1}, $TC_{k-2}, \ldots$, and TC_0 can be used to activate this type of faults. The outputs of the faulty cell will no longer be a complete sequence and can be observed from the cell's outputs. We conclude that all the multiple access faults can be detected with 100% fault coverage. □

Testing of D Flip–Flops

D flip–flops are tested using functional model. That is, DFFs are treated as pipelined latches. A fault may cause the flip–flop to be unable to receive data or be triggered by the correct clock edge, or to be incapable of being set or reset. With our configurations TC_0 through TC_k, there exists at least one configuration, which contains the sequences 010 and 101. It should be noted that the configuration memory must be configured in order to direct the outputs from the LUT through the highlighted path to the CLB's output. Fortunately, this will not increase extra configurations and can be done during the testing of LUTs. Therefore, DFFs can be used as pipelined latches and could be fully tested using the same configurations and TPs as for an LUT.

Testing of Multiplexers

As the fault model defined above, stuck-on and stuck-off faults are assumed. For stuck-off faults, we have assumed that the output is either stuck-at 1 or stuck-at 0 and can be collapsed as stuck-at faults at the output line of an LUT. This type of faults can be simultaneously tested as the testing of an LUT according to Theorem 1. For stuck-on faults, wired-operations are performed and the faults can be detected in the same manner as described in Theorem 2.

Now that we have proven the testability conditions for a single cell, we now turn to the problem of testing a whole array. The first problem is how to send the TPs to each cell in an FPGA (controllability) and propagate the fault effects to the primary outputs (observability). The controllability and observability problems can be solved using the following theorem.

THEOREM 3 For an FPGA cell array, all the faults defined can be tested with a minimal complete input sequence and configurations $TC_k, TC_{k-1}, \ldots, TC_0$ for each cell.

Proof As shown in Theorem 1 and Theorem 2, a complete input sequence and test configurations TC_i, $0 \leqq i \leqq k$ can be used to test all the faults within a cell. For an entire cell array, we must solve the controllability and observability problems. First, all the cells must have the same configurations during each test phase, i.e. they are configured to the same bijective cell function. Since all the linear arrays are identical, we connect all the inputs of the left-most cells to IOBs and an external TG is used to generate TPs for all the arrays as shown in Fig. 8. In this figure, since $k - 1$ inputs of each cell just pass through the cell, therefore, these $k - 1$ inputs are extracted as external signal buses. Since all the input IOBs also receive minimal complete input sequence, the faults that occur in an IOB can also be detected. The output IOBs can also be tested in the same manner. The input to each linear array is a minimal complete sequence, the simplest approach to implement the TG is by using a *k*-bit binary counter, which counts from 0 to $2^k - 1$.

As we know, a minimal complete input sequence is sufficient to completely test a cell. Let θ be a minimal complete input sequence. We apply θ to the input of the leftmost cell of each linear array. Since each cell is configured as a bijective cell function, then if $cell_{00}$ is fault-free, the output sequence θ, is also a minimal complete sequence. This sequence can then be used as

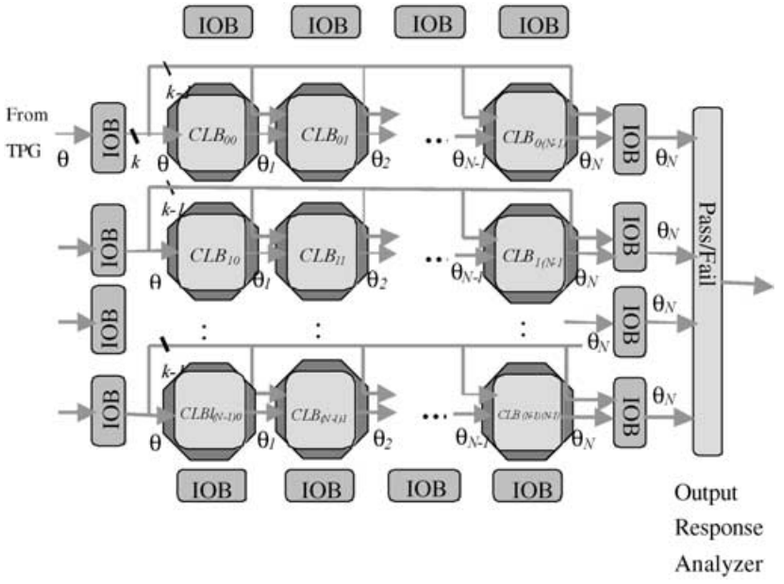

FIGURE 8 Application of test patterns.

the input sequence for cell_{01}, obtaining complete sequence θ_2. We then apply θ_2 to cell_{02}, obtaining complete sequence θ. Reiterating this process, we construct a complete input sequence for each cell in the linear array. The same mechanism can be applied to the other linear arrays. This solves the controllability problem.

Now we turn our attention to the observabiltiy problem. If cell_{00} is faulty, a faulty effect must occur at the outputs of this cell. Then, since the cell function is bijective, the faulty effect will be propagated through cell_{01} and appears at its outputs. This is because any input change is propagated to an output change. Bijectivity ensures that the change continues to ripple to the outputs of the rightmost cell, which in turn, connected to IOBs and output response analyzer for observation. The output sequences of all linear arrays are identical if no faults exist in the FPGA chip. The output response analyzer is simply a comparator and can be used to compare these outputs and produce a pass/fail indication. Other types of analyzers can also be used. For example, a checksum checker can be used to check the sum of the output sequence. Since we assume that there exists at most one faulty cell in a linear array, the propagating path cannot mask each fault. We conclude that the cell faults in each linear array can be detected by the complete input sequence.

An additional configuration must be added to test the c input of MUX1. With this configuration, the output of the LUT is forwarded to the input of MUX1. The c input is propagated through the highlighted path as shown in Fig. 2 to the output y and used as input for the neighboring cell. Therefore, the number of configurations is $k+2$ to completely test the whole FPGA array. □

BUILT-IN SELF-TEST

The BIST structure is presented in this section. A group of CLBs is configured as test patterns generators (TPGs) and some as output response analyzers (ORAs), all other cells are cells under test (CUT). Our BIST structure contains two test sessions (*horizontal* and *vertical* sessions) as shown in Figs. 9 and 10. The TPG works as a binary counter to provide minimal complete input sequence for the leftmost cells. Since a CLB has only one output line, therefore, k CLBs are necessary to build a k-bit binary counter. The output analyzer can be implemented with a checksum analyzer or comparator. Details of our BIST structures can be found in our previous paper [18].

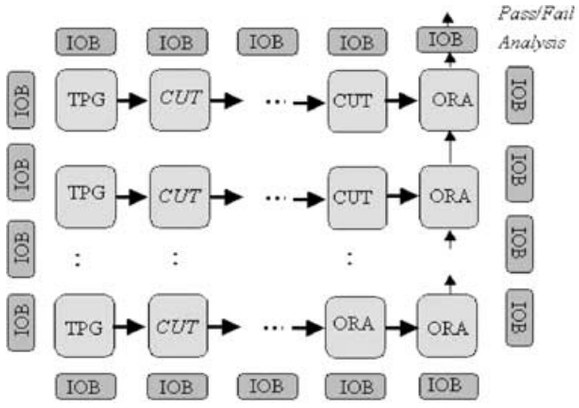

FIGURE 9 BIST structure for a BIST session.

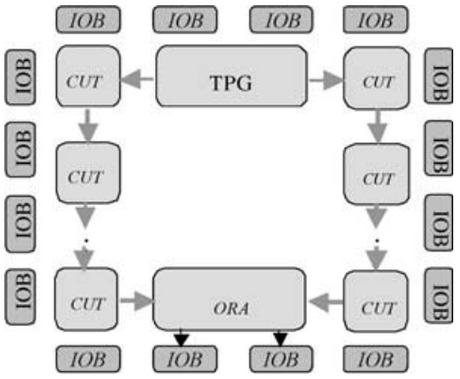

FIGURE 10 Second BIST session.

FAULT DIAGNOSIS

In this section, we introduce the ILA-based approach for diagnosing an FPGA that fail the test provided by the two BIST sessions depicted above. In the same way, fault diagnosis approaches also falls into two ways; one is fault diagnosis for unprogrmmed FPGAs, and the other is that for programmed FPGAs. We focus on fault diagnosis for unprogrammed FPGAs. If a faulty part in an FPGA can be identified prior to programming it, by isolating the faulty part, we can implement a required logic on the FPGA using only the fault-free part. Our fault diagnosis process contains two sessions: (1) horizontal diagnosis, and (2) vertical diagnosis. These two steps are described as follows.

Session 1 (Horizontal Diagnosis): Horizontal diagnosis is illustrated in Fig. 11. The outputs of the rightmost cells are compared with correct results for identifying faulty rows.

Session 2 (Vertical Diagnosis): The interconnection structures for vertical diagnosis is shown in Fig. 12. The outputs of the bottommost cells are compared with correct results for identifying faulty columns. By intersecting faulty columns with faulty rows, the faulty cells can be identified. The diagnosis resolution of our approach is a single CLB due to the structure of a basic cell. However, with the approach proposed in Ref. [18], three diagnosis sessions are required for diagnosing a faulty CLB. Therefore, the proposed approach can reduce the number of test configurations and the diagnosis speed can be improved significantly.

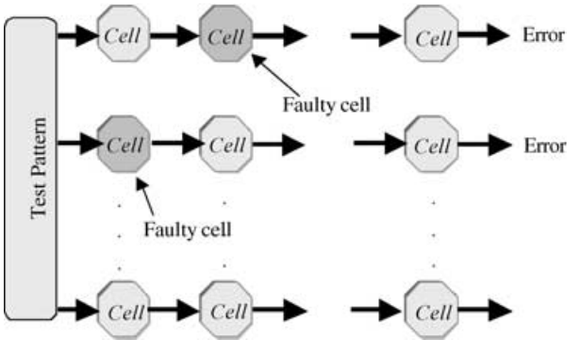

FIGURE 11 Horizontal diagnosis.

DIAGNOSIS COMPLEXITY ANALYSIS

DEFINITION The *diagnosis complexity* is defined as the time complexity required to perform fault diagnosis for an FPGA.

DEFINITION An FPGA is *C-diagnosbale* if there exists a fault diagnosis approach, whose complexity is independent of the FPGA's array size.

In order to ease the analysis of complexity for fault diagnosis, some terminology and notations used in Ref. [6] are still used here. The *fault diagnosis procedure (FDP)* for an FPGA can be expressed as

$$\mathrm{PDP} = [(\mathrm{TC}_0, S_0), (\mathrm{TC}_1, S_1), \ldots, (\mathrm{TC}_{\mathrm{nc}}, S_{\mathrm{nc}})],$$

where S_i, $0 \le i \le \mathrm{nc}$, denotes the minimal complete input sequence applied for test configuration TC_i. According to previous discussions, we can see that

$$|S_0| = |S_1| = \cdots |S_{\mathrm{nc}}| = 2^k.$$

Moreover, the number of configurations $\mathrm{nc} = 2(k + 2) = O(\log n)$ for fault diagnosis. Let t_c denote the time required to load one bit of a program into a CMC. The time required to implement all test configurations then can be expressed as

$$T^C(\mathrm{FDP}) = \sum_{i=0}^{\mathrm{nc}-1} t_c c(i),$$

where $c(i)$ denotes the number of CMCs to implement configuration TC_i. Since each CLB implements the same logic function for each configuration, we can treat it as an iterative array system. Therefore, the block-sliced loading approach [10] can be used. Therefore, the time required

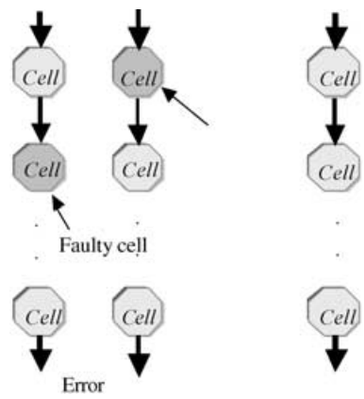

FIGURE 12 Vertical diagnosis.

TABLE I Comparison with other works

Approach	Goal	NTP	NC	NCD	TSBIST
Inoue [6]	Fault diagnosis	2^k	$2k+4$	$4k+8$	–
Stroud [9]	Built-in self-test	2^k	–	–	3
ILA-based fault detection	Fault detection, BIST, and diagnosis	2^k	$k+2$	$3k+6$	2
Hybrid fault detection	Fault detection, BIST, and diagnosis	2^k	$k+2$	$2k+4$	2

TABLE II Complexity of FDTP

FDTP	Universal test procedure [10]	Universal fault diagnosis [5]	Fault diagnosis [6]	This paper
Complexity	$O(n\log^2 n+\log^3 n)$	$O(N^2 n\log n)$	$O(n\log n)$	$O(n\log n)$

to implement configurations is

$$T^C(\text{FDP}) = \sum_{i=0}^{\text{nc}-1} t_c c(i) = O(n\log n).$$

Furthermore, the time required to apply all the input sequences in FDP is

$$T^S(\text{FDP}) = \sum_{i=1}^{n_c} t_s s(i) = 2^k \sum_{i=1}^{n_c} t_s, = O(n)$$

where t_s denotes the clock cycle time. The diagnosis complexity of our approach then can be expressed as

$$T(\text{FDP}) = T^C(\text{FDP}) + T^S(\text{FDP}) = O(n\log n).$$

Note that the diagnosis complexity is independent of the array size. Therefore, we can obtain C-diagnosable FPGAs. We compare our approach with Refs. [6,9], since the same CLB structure is used. The result is shown in Table I. In this table, we list the goals of these papers, and compare the number of TPs for each configuration (NTP), the number of configurations for fault detection (NC), the number of configurations for fault diagnosis (NCD), and the number of test sessions for BIST (TSBIST). The number of test configurations for fault detection (NTC) is only half of that in Ref. [6]. Moreover, the number of BIST sessions is also less that that in Ref. [9].

The complexity for fault diagnosis of our approach is shown in Table II and compared with that in Refs. [5,6,10]. From this table, we can see that our approach is better than Refs. [10] and [5]. Although the diagnosis complexity of our approach is the same as that in Ref. [6], however, if the constant items are considered as shown in previous discussions, our approach is also better than Ref. [6].

CONCLUSIONS

A novel approach for testing FPGAs based on LUTs is proposed in this paper. Each CLB in an FPGA is configured as a cell with k inputs and outputs. We use a cell as the basic test element instead of using a CLB. The whole chip is partitioned into disjoint one-dimensional arrays of cells. We assume that in each linear array, there is at most one faulty cell, and multiple faulty cells existing in different arrays can also be detected.

For a faulty LUT, a fault may occur at the memory matrix, decoder, input and output lines. Stuck-on and stuck-off fault models are adopted for multiplexers. Moreover, we assume that the interconnection network has been tested. Our idea is to configure each cell function *bijective*. This bijective cell function is helpful for applying pseudo-exhaustive TPs to each cell under test and propagating errors through the cell arrays. We require $k+2$ configurations to test all the faults defined. The number of configurations is less than previous works and the test time can be significantly reduced. Experimental results show that 2^k TPs are sufficient for pseudo-exhaustively testing the chip and the resulting fault coverage is 100%.

This BIST structure (test pattern generator and output response analyzer) is easy to implement. The routing complexity and the required number of IOBs are reduced significantly. To diagnose a faulty CLB, two diagnosis sessions are required. Diagnosis complexity is also analyzed and compared with previous works. The result shows that C-diagnosable FPGAs can be obtained. Although the diagnosis complexity of our approach is the same as that in Ref. [6], however, if the constant items are considered, our approach is better than that in Ref. [6].

References

[1] Brown, S., Francis, R.J., Rose, J. and Vranesic, Z.G. (1992) Field Programmable Gate Arrays (Kluwer Academic, Boston, MA).
[2] Lu, S.K., Shih, J.S. and Wu, C.W. (1999) "Testing configurable LUT-based FPGAs", Proc. 10th VLSI/CAD Nantou Taiwan, pp. 171–174.
[3] Quddus, W., Jas, A. and Touba, N.A. (1999) "Configuration self-test in FPGA-based reconfigurable systems", Proc. IEEE International Symposium on Circuits and Systems (ISCAS), (Orlando), pp. I-97–I-100.
[4] Huang, W.K., Meyer, F.J., Chen, X.T. and Lombardi, F. (1998) "Testing configurable LUT-based FPGAs", *IEEE Trans. VLSI Systems* **6**(2), 276–283.
[5] Inoue, T., Miyazaki, S. and Fujiwara, H. "On the complexity of universal fault diagnosis for lookup table FPGAs", In *Proc. Asain Test Symposium 1997 (ATS'97)*, pp. 276–281.

[6] Inoue, T., Miyazaki, S. and Fujiwara, H. (1998) "Universal fault diagnosis for lookup table FPGAs", *IEEE Design and Test of Computers*, pp. 39–44.
[7] Huang, W.K., Meyer, F.J., Chen, X.T. and Lombardi, F. (1996) "A general technique for testing FPGAs", *Proc. IEEE VLSI Test Symp.*, (Princeton, NJ), pp. 450–455.
[8] Liu, T., Huang, W.K. and Lombardi, F. (1995) "Testing of uncustomized segmented channel FPGAs", *Proc. ACM Symp. FPGAs*, pp. 125–131.
[9] Stroud, C., Chen, P., Konala, S. and Abramovici, M. (1995) "Using ILA testing for BIST in FPGAs", *Proc. IEEE VLSI Test Symp.*, pp. 256–261.
[10] Inoue, T., Fujiwara, H., Michinishi, H., Yokohira, T. and Okamoto, T. (1995) "Universal test complexity of field programmable gate arrays", *Proc. Fourth IEEE Asian Test Symp.* (IEEE CS Press), pp. 259–265.
[11] Renovell, M., Portal, J.M., Figueras, J. and Zprian, Y. (1998) "Testing the Interconnect of RAM-Based FPGAs", *IEEE Design and Test of Computers*, pp. 45–50.
[12] Michinishi, H., Yokohira, T., Okamoto, T., Inoue, T. and Fujiwara, H. (1996) "A test methodology for interconnect structures of LUT-based FPGAs", *Proc. Fifth Asian Test Symp.* (IEEE CS Press), pp. 68–74.
[13] Stroud, C., Konala, S., Chen, P. and Abramovici, M. (1996) "Built-In self-test of logic blocks in FPGAs (Finally, a free lunch: BIST without overhead!)", *Proc. IEEE VLSI Test Symp*., pp. 387–392.
[14] Stroud, C., Lee, E., Konala, S. and Abramovici, M. (1996) "Selecting built-in self-test configurations for field programmable gate arrays", *Proc. IEEE Automatic Test Conference*.
[15] Stroud, C., Lee, E., Konala, S. and Abramovici, M. (1997) "BIST-based diagnostics of FPGA logic blocks", *Proc. IEEE International* Test Conference, pp. 539–547.
[16] Itazaki, N., Matsuki, F., Matsumoto, Y. and Kinoshita, K. (1998) "Built-In self-test for multiple CLB faults of an LUT type FPGA", *Proc. Asian Test Symposium*, pp. 272–277.
[17] Programmable Gate Array Data Book, Xilinx, Inc., San Jose, CA, 1991.
[18] Lu, S.K. and Shih, J.S. (2000) "A novel approach to testing LUT-based FPGAs", *Journal of Information Science and Engineering*, 733–750.
[19] Wu, C-F. and Wu, C-W. (1999) "Fault detection and location of dynamic reconfigurable FPGAs", *Intl. Symp. VLSI Technol., Syst., Appl.*, 215–218.

Shyue-Kung Lu received his PhD (1996) in electrical engineering from the National Taiwan University. From 1996 February to 1998 August, he was an associate professor in the Department of Electrical Engineering at Lunghwa Junior College of Technology and Commerce. Since then, he has been with the Department of Electronic Engineering at Fu Jen Catholic University, where he is an associate professor. His research interests include the areas of VLSI testing and fault tolerant computing.

Fu-Min Yeh received his PhD (1997) in Electrical Engineering from the National Taiwan University. He is a member of technical staff at Chung-Shan Research Institute of Science and Technology. His research interests include VLSI testing and fault-tolerant computing.

Jen-Sheng Shih is a postgraduate of electronic engineering at Fu Jen Catholic University, Taiwan. Mr Shih received a diploma in Electronic Engineering from Tungfang Junior College of Technology, Taiwan, in 1993, and the Bachelor's degree in electronic engineering from National taipei University of Technology, Taiwan, in 1998. He conducts his master's thesis research on VLSI testing under the supervision of S.-K. Lu.