To appear in: *INTEGRATION the VLSI Journal*

On the Testability of Iterative Logic Arrays

B. Becker*, R. Hahn*, J. Hartmann†, U. Sparmann†,

Technical Report 4/1994
SFB 124–B1

* Fachbereich Informatik, Johann Wolfgang Goethe–Universität,
D 60054 Frankfurt am Main.
Email: <name>@kea.informatik.uni-frankfurt.de

† Fachbereich 14.1 – Informatik, Universität des Saarlandes,
D 66041 Saarbrücken.
Email: <name>@cs.uni-sb.de

On the Testability of Iterative Logic Arrays*

Bernd Becker Ralf Hahn
Computer Science Department, J.W.G.-University
D-60054 Frankfurt am Main, Germany
email: <name>@kea.informatik.uni-frankfurt.de†

Joachim Hartmann‡ Uwe Sparmann‡,§
Computer Science Department, University of Saarland
D-66041 Saarbrücken, Germany
email: <name>@cs.uni-sb.de

Abstract

The problem of detecting single cellular faults in arbitrarily large one-dimensional, unilateral, combinational iterative logic arrays *(= ILAs) is considered.* Fault patterns *(= FPs) of the ILA's basic cell are introduced to characterize any cellular fault. Testability properties like (full, partial) testability, redundancy, test complexity of FPs are studied. Based on this analysis we prove that only two* test complexity *classes exist: The minimum size of a complete test set of an ILA is either constant - in this case the ILA is called* C-testable *- or linear in the length of the ILA. Furthermore, depending on the type of the FP that is considered new efficient algorithms for the determination of the test complexity and the construction of complete test sets are presented. In particular, we reduce the exponential worst case complexity of the construction given in [1] to a polynomial worst case bound (measured in the size of the function table for the basic cell).*

*A preliminary version of this paper appeared in the Proc. of the MFCS 92, LNCS 577, pp. 142–152
†B. Becker and R. Hahn were supported by DFG, Grant Be 1176/3-2
‡Research supported by DFG, SFB 124–"VLSI Entwurfsmethoden und Parallelität"
§Work partly done while visiting the University of Iowa supported by DFG, Grant No. Sp431/1-1.

1 Introduction

Iterative arrays are becoming increasingly popular with the advent of very large scale integration (VLSI) and Wafer Scale Integration (WSI). The manufacturing of large array systems for use e.g. in arithmetic logic units, parallel multipliers and dividers, signal processing, pattern recognition, bit-sliced systems ([2, 3, 4]) is now manageable. Advantages such as simplified design, layout and testing have motivated research on the capabilities of ILAs since the beginning of the sixties [5, 6, 7, 8, 9, 10]. We focus on testability aspects of iterative structures in this paper. Before going into the details we want to emphasize the practical relevance of testing problems by making some general remarks. For a comprehensive treatment of the topic see e.g. [11].

As a result of technological improvements, VLSI electronic circuitry may contain hundreds of thousands of transistors on a single silicon chip. Even if the chips are correctly designed, a non negligible fraction of them will have physical defects caused by imperfections occurring during the manufacturing process. Therefore, there has to be a test phase in which the "good" chips are sorted from the "bad" ones. Because of the variety of possible defects restrictions on a subset of the possible faults are necessary; these simplifying assumptions based on the experience of many years are manifested in fault models. Since tests are generated to test for the fault mechanisms described by the assumptions, the reliability of the chip is at least partially determined by the accuracy and effectiveness of the fault model (measured e.g. in detected physical failures). The classical stuck-at fault model is well-known and used throughout the industry. It assumes that a defect causes a basic cell input or ouput to be fixed to either 0 or 1. Thus all failures with this effect will be detected by tests for stuck-at faults. It has been observed by many authors that a large amount of defects typical for modern VLSI technologies are not covered by stuck-at faults [12, 13, 14, 15]. Other fault models are necessary to overcome this problem, e.g. flexible fault models which verify the correct static behaviour of a combinational circuit based on inductive fault analysis [16, 17] or dynamic fault models which allow to model stuck-open faults or timing issues [18, 19, 20]. The strongest cell-based fault model to check the correct static behaviour of a combinational circuit is the cellular fault model (CFM), which tries to completely verify the function of each basic cell in the circuit [1, 21]. The results presented in the following are based on this fault model. Furthermore, it is as usually assumed that there is at most one fault (according to the CFM) in the circuit (= *single fault assumption*).

Inspite of these restrictions the test generation problem for general Boolean circuits remains *NP*-complete from the theoretical point of view. This relates to the following observations in practice. With increasing complexity of VLSI circuits, the costs for the test phase have risen dramatically, at least 25% and up to 60-70% of the total product costs count on testing [22, 23]. Specialists in the field of testing agree that testability issues have to be considered from the very beginning of the design process to control the test costs and to guarantee the testability of the circuit at the end of the manufacturing process. This requires a detailed analysis of the relation between structural properties and testability properties of the circuit being constructed. Relevant subclasses of circuits which, due to their structural porperties, allow efficient test generation algorithms and as a result yield complete test sets of small size should be classified and investigated. Typical examples of classes which have been successfully treated by this method are PLAs, memories, arithmetical units and tree-like structures (see e.g. [24, 25, 26, 27, 21, 28, 29]).

In the context of this paper we restrict our attention to iterative array structures. As indicated at the beginning testability aspects of iterative logic arrays of several differing types (one-dimensional, two-dimensional, unilateral, bilateral, orthogonal, combinational, synchronized, homogeneous, inhomogeneous) have been studied in the literature. The first paper in this context authored by Kautz appeared in 1967 [6] and intitialized research activities which led to numerous extensions and variations of the results presented there. For a good overview see e.g. [10, 30, 4, 9]. Common to almost all these papers is the following approach: under varying assumptions sufficient conditions to ensure *C-testability*, i.e. complete testability with a number of tests that only depends on the basic cell types of the array and is independent of the overall size of the array, are derived. If an array does not fulfill these properties, modifications of the basic cells are proposed to guarantee C-testability (see e.g. [1, 31, 4]). Of course, these modifications must not destroy the original behaviour of the cells and should cause minimal overhead. The sufficient conditions given are directed to allow modifications with small overhead, but normally they are not necessary, i.e. there are ILAs which are modified to become C-testable, although they are C-testable at the beginning. In general, it turns out to be very difficult to characterize C-testable ILAs by necessary and sufficient conditions. For some classes of two-dimensional arrays elementary problems in the test generation process such as the controllability problem even turn out to be undecidable, if they are to be solved uniformly for all arrays in the class [5]. We show that problems of this type do not occur if one considers one-dimensional arrays.

More precisely, it is the purpose of this paper to point out that the testability properties of one-dimensional, horizontally connected, unilateral, homogeneous, combinational ILAs can be completely characterized. (From now on the term ILA is to be interpreted in this sense. Especially notice that only one basic cell type is used to build up the ILAs.) Our investigations are a continuation of the fundamental work done by Kautz [6] and later by Friedman [1]. We shortly review the main results of both papers and then come to the extensions obtained in this paper. Kautz and Friedman as well make the assumption that the ILAs under test are fully testable, i.e. redundant faults do not occur. Kautz then characterized the ILAs which are testable with NM input patterns. NM is the total number of input patterns to the basic cell considered and, since in a fully testable ILA all possible input combinations have to be applied to each cell, this is the optimal size of a complete test set. Friedman noticed that there are ILAs which are not testable with NM tests but with a complete test set whose size only depends on the basic cell, but is independent of the length of the circuit under test. He was able to give necessary and sufficient conditions for C-testability under the assumptions that full testability is given and the basic cell has no vertical outputs.

In this paper we add the following results:

First, the assumption of full testability is dropped, i.e. redundancies are allowed and have to be considered. Whereas the detection of redundancies is *co-NP*-complete for the general case, we prove that in ILAs redundancies can be completely characterized by inspection of the basic cell. This result and the subsequent investigations as well make use of fault patterns (FPs), which characterize any cellular fault and significantly simplify the handling of the fault model. Next, the test complexity classes of ILAs are determined. For ILAs without vertical outputs the results of Friedman can be used to show that there exist only two complexity classes, the C-testable ILAs and ILAs with worst, i.e. linear, test complexity. We demonstrate that this "gap-theorem" remains valid if some of the cells in an ILA contain redundant cellular faults and basic cells with vertical outputs are allowed. It follows that algorithms for deciding C-testability together with construction methods for complete C- and linear tests, respectively, provide an entire asymptotically optimal solution of the testability problem for the considered class of circuits.

We then turn to the question how such algorithms can be obtained and how their complexity can be characterized. The complexity of the algorithms is related to the size $N \cdot M$ of the function table for the basic cell. (In particular, this implies that the time complexity of the algorithms is independent of the size of the ILA, even if we have an ILA with linear test complexity.) Also the size of the C-tests constructed is measured in terms of N and M. (Notice, that a C-test only has constant size compared to the length of the ILA, whereas it might even be exponentially larger than the function table of the basic cell.)

As mentioned before FPs are used to construct the tests. Five different cases are distinguished depending on the properties of the FP that is considered. Case 1 deals with partially testable FPs, Cases 2-4 handle different types of C-testability, Case 5 is concerned with non C-testable FPs. Cases 1 and 5 can be solved by simple $O(N^2M)$ graph algorithms on the *state transition graph* and the *pair graph* of the basic cell. For the construction of C-tests in Cases 2-4 we generalize the concept of *repetition sequence* to our settings (vertical outputs, redundancies allowed). Similar to the original paper by Friedman, the existence of a repetition sequence is a necessary and sufficient condition for C-testability and the construction of a repetition sequence R directly yields a *repetitive C-test* of size $|R| + 1$. (Thus, a repetitive C-test has minimal size, if the repetition sequence is as short as possible.) We give algorithms to determine repetition sequences in all cases. The time complexity of these algorithms is bounded by $O(N^4M)$ and the size of the C-tests is bounded by $O(N^3)$. This reduces the exponential worst case complexity of the construction given in [1] to a polynomial worst case bound. Nevertheless, it turns out that the repetitive C-tests constructed by our algorithms in general are not minimal. (Minimality is only obtained for Cases 2 and 3.) We prove that minimization of the repetition sequence in Case 4 is an *NP*-hard problem and therefore an exact solution is computationally (too) expensive. In this sense our algorithm is a reasonable compromise between small test size and efficiency of computation.

The paper is structured as follows: In Section 2 we provide necessary definitions and notations from the field of iterative logic arrays and testing. (k-parallel) state transition graphs are introduced to model the behaviour of ILAs in the presence of faults. Fault patterns are defined and their relation to ordinary cellular faults is discussed. Our results on the testability of ILAs are given in Section 3. A detailed analysis of fault patters is used to derive algorithms for the determination of the test complexity and for the generation of complete test sets for single FPs and ILAs as well. We finish with a discussion of the results and mention some open problems.

2 Preliminaries

We start with a short description of definitions and notions necessary for the purpose of this paper. More general information on iterative logic arrays and testing is for example found in [10].

Iterative logic arrays (ILAs) are circuits which are composed of copies of one or a small number of basic cell types C. As mentioned before, in general, there is a large variety of different ILA-types depending on the properties of the basic cell and the overall composition rule. For this paper we only consider basic cells C with a *combinational* behaviour and an *unilateral* signal flow from the west and north to the south and east of C (see Figure 1). Composition is allowed in west-east direction

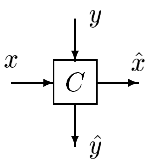

Figure 1: Basic cell C

(see Figure 2), the resulting ILAs are called *one-dimensional, unilateral, combinational* ILAs *over the basic cell* C. We use the term ILA in this sense for the rest of the paper. Let $C(n)$ denote the ILA composed of n copies of C. n is called the *length* of $C(n)$. The cells in $C(n)$ are numbered from 1 to n starting with 1 at the leftmost cell. C_i denotes the cell with number i.

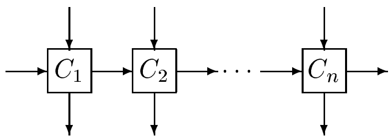

Figure 2: ILA over C of length n

The behaviour of $C(n)$ is inductively defined by the behaviour of C, which we assume is given as follows. X with $\#X =: N > 0$ (Y with $\#Y =: M > 0$) is the set of possible input values in the west (north) of C. $x \in X$ ($y \in Y$) is also called *x-input* (*y-input*). The set of possible output values in the east (= *x-outputs*) is a non empty subset of X, the set of output values in the south (= *y-outputs*) is a set $\hat{Y}$ which may even be empty. The behaviour of C then is a function

$$g := g^X \times g^Y : X \times Y \to X \times \hat{Y}$$

which defines the x-output $\hat{x} = g^X(x, y)$ and the y-output $\hat{y} = g^Y(x, y)$ for each input combination $(x, y) \in X \times Y$. (Notice that the inputs and outputs in the north, south, east and west of C are not necessarily binary, but may correspond to busses of binary input and output lines.) g^X can be generalized to a function with domain $X \times Y^*$ instead of $X \times Y$ by setting $g^X(x_1) := x_1$ and $g^X(x_1, y_1, \ldots, y_i) := g^X(g^X(x_1, y_1, \ldots, y_{i-1}), y_i)$ for $i \geq 1$. Now, consider $I = (x_1, y_1, \ldots, y_m) \in X \times Y^*$ with $m \geq n$, then I naturally defines a primary input combination for $C(n)$, namely $I_n := (x_1, y_1, \ldots, y_n)$.

$$\begin{aligned}
&g^Y_{(i,j)}(I) := \\
&\qquad (g^Y(g^X(x_1, y_1, \ldots, y_{i-1}), y_i), \\
&\qquad g^Y(g^X(x_1, y_1, \ldots, y_i), y_{i+1}), \ldots \\
&\qquad g^Y(g^X(x_1, y_1, \ldots, y_{j-1}), y_j))
\end{aligned}$$

denotes the y-outputs of $C_i, C_{i+1}, \ldots C_j$, when I is applied to $C(n)$. All in all, the y-outputs from C_1 to C_n together with the x-output at C_n provide the primary output combination of $C(n)$ resulting from the application of I, i.e. the *function $g_{C(n)}$ of $C(n)$* is given by

$$g_{C(n)}(I) := (g^Y_{(1,n)}(I), g^X(I_n))$$

As an example for the type of ILA defined take a fulladder and the resulting ILA, a carry ripple adder. In general, any finite automaton with output defines a cell C.

In the following it will be helpful to model the behaviour of C and ILAs over C by using the interpretation of C as automaton and defining *(k-parallel) state transition graphs.* For this X is the set of states and $t := (x, y) \to (\hat{x}, \hat{y})$ for $(x, y) \in X \times Y$ and $(\hat{x}, \hat{y}) = g(x, y)$ is called a *transition.* Then for $k \in \mathbf{N}$ the k-parallel state transition graph contains edges corresponding to k parallel transitions under a fixed y-input y. Formally, the *k-parallel state transition graph of C* is defined as $G^k_C := (X^k, E_k)$. E_k, the set of edges, contains a labeled edge $e_{(t_1, t_2, \ldots, t_k)}$ for each tuple of k transitions $t_i = (x_i, y) \to (\hat{x}_i, \hat{y}_i)$. The source of $e_{(t_1, \ldots, t_k)}$ is state tuple $(x_1, \ldots, x_k)$, the sink is state tuple $(\hat{x}_1, \ldots \hat{x}_k)$, the label of $e_{(t_1, \ldots t_k)}$ is input y and output tuple $(\hat{y}_1, \ldots, \hat{y}_k)$. The size of G^k_C denoted by $\#G^k_C$ is $N^k M$. The cases $k = 1$, $k = 2$ and $k = 3$ will be of major importance in this paper. $G^1_C =: G_C$ (G^2_C) is called the *state transition graph (pair graph) of C.* An edge $e_{(t_1, t_2)}$

in the pair graph is *x-distinguishing (y-distinguishing)*, if a difference in the states x_1, x_2 causes a difference at the x-outputs (y-outputs), i.e. if $\hat{x}_1 \neq \hat{x}_2$ ($\hat{y}_1 \neq \hat{y}_2$).

Testability properties of ILAs are the topic of this paper. We measure testability with respect to a fixed fault model, the single cellular fault model. It is assumed in the single cellular fault model (CFM) that a fault F modifies the correct behaviour g of exactly one cell C_i of an ILA $C(n)$ to a combinational function $g_{faulty} = (g^X_{faulty}, g^Y_{faulty})$. i is called the *fault location* of F. Let $g_{C(n),F}$ be the function of $C(n)$ in the presence of F. Then a primary input combination $I = (x_1, y_1, \ldots, y_n)$ is a *test* for F, if $g_{C(n),F}(I) \neq g_{C(n)}(I)$. More precisely, this means the following:

i) I *generates* values x_i and y_i at C_i with $x_i := g^X(x_1, y_1, \ldots y_{i-1})$ and $g_{faulty}(x_i, y_i) \neq g(x_i, y_i)$.
ii) The *difference* resulting at C_i is *propagated* to a primary output line of a cell C_{i+k} of $C(n)$ with $0 \leq k \leq n - i$. Here, two cases are possible. If $g^Y_{faulty}(x_i, y_i) \neq g^Y(x_i, y_i)$, the difference $g^Y(x_i, y_i)/g^Y_{faulty}(x_i, y_i)$ is visible in the south of C_i. Otherwise the output states at C_i $g^X(x_i, y_i) =: x_{i+1}$ and $g^X_{faulty}(x_i, y_i) =: x_{i+1,f}$ are *distinguished* by I, i.e. $g_{C(n-i)}(x_{i+1}, I') \neq g_{C(n-i)}(x_{i+1,f}, I')$ for $I' := (y_{i+1}, \ldots, y_n)$.

Notice that the testability of F by I can directly be translated to the existence of paths with certain properties in the graphs G_C and G^2_C.

F is *testable*, if a test exists, otherwise F is *redundant*. For $n \in \mathbf{N}$ let $F(n)$ be a set of cellular faults in $C(n)$. If each cellular fault in $F(n)$ is testable, then $F(n)$ is *fully testable*. The notions *partially testable* or *fully redundant* apply, if some of the faults are testable or if each fault is redundant. $C(n)$ is *fully testable*, if each cellular fault in $C(n)$ with respect to the CFM is testable. If $F(n)$ is fully testable for each $n \in \mathbf{N}$, then $F(\cdot)$ denoting the collection of fault sets $F(n)$ is called *fully testable*. If $C(n)$ is fully testable for each $n \in \mathbf{N}$, then $C(\cdot)$, the *set of ILAs over C*, is called *fully testable*. If full testability cannot be obtained, the test generation process should at least yield a *complete test set*, i.e. a test set that contains a test for each testable fault in the considered fault set. It follows directly that the construction of complete test sets requires the determination of the redundant faults and the construction of tests for the testable faults. Note that full testability is decided implicitly during the determination of the redundancies.

A widely accepted criterion for the quality of a complete test set is the size of the set, since it determines the test application time. We therefore define the *test complexity* of a fault set (of an ILA) as the number of elements in a minimal test set for this fault set (for this ILA). The *test complexity of $F(\cdot)$* is defined as a function TC_F from $\mathbf{N} \to \mathbf{N}$ with

$$TC_F(n) := \min\{\#\mathcal{T} \mid \mathcal{T} \text{ is complete for } F(n)\}$$

If $F(n)$ is the set of all cellular faults in $C(n)$ we get the *test complexity of $C(\cdot)$* denoted by the function TC_C. The determination of TC_C is one of the objectives in this paper. The following is clear in advance: since there are at most $O(n)$ different cellular faults in an ILA of length n and at least one cellular fault is testable, we get a linear upper bound $TC_C(n) = O(n)$ and a constant lower bound $TC_C(n) = \Omega(1)$ for the test complexity of $C(\cdot)$. (The upper bound can easily be refined to $NM((N-1)(n-1)+1)$ (see also [1]).) ILAs over C (fault sets $F(\cdot)$) are called *C-testable*, if $TC_C(n) = \Theta(1)$ ($TC_F(n) = \Theta(1)$), i.e. a complete test set with a number of tests which is independent of n, the number of cells in the ILA, exists. C-testable ILAs are ILAs with optimal test complexity. ILAs with (asymptotically) worst test complexity are ILAs which require a linear number of tests, i.e. $TC_C(n) = \Theta(n)$.

The determination of the test complexity and the construction of complete test sets is carried out with the help of *fault patterns* (FPs). FPs implicitly define (sets of) cellular faults, whose testability properties turn out to be significant for the testability properties of the ILA as a whole. In the remaining part of this section we introduce FPs and point out some of the relations between FPs and the set of all cellular faults.

A fixed FP describes a behaviour of C which differs from the correct behaviour g in a prescribed manner for exactly one input combination. To be more precise, consider a transition $t = (x, y) \to (\hat{x}, \hat{y})$ of C and define exactly one *y-fault pattern* (yFP) and $N-1$ *x-fault patterns* (xFP). The yFP for t is given by $x^{y}_{\hat{y}/*}$. A yFP together with a location i defines a *simple* cellular fault with faulty behaviour at cell C_i: the y-output for input combination (x, y) is faulty, all other input combinations at C_i generate correct results. An xFP for t is defined by $x^y\hat{x}/\hat{x}_f$ ($\hat{x}_f \in X$ with $\hat{x}_f \neq \hat{x}$) and indicates a faulty x-output $\hat{x}_f$ for input combination (x, y). Clearly, an xFP together with a location i defines a *simple* cellular fault whose faulty behaviour is completely given by a faulty x-output $\hat{x}_f$ at C_i provided that input combination (x, y) is applied to C_i. Notice that in general a cellular fault F is given by a function g_{faulty} at C_i with faulty outputs for possibly more than one input combination and thus F is *composed* from several FPs with common fault location i. A yFP $x^{y}_{\hat{y}/*}$ (xFP $x^y\hat{x}/\hat{x}_f$) is a *component of F at* C_i, if $g^Y_{faulty}(x, y) \neq \hat{y}$ ($g^X_{faulty}(x, y) = \hat{x}_f$).

For each $n \in \mathbf{N}$ an FP (given without fault location) defines a fault set $F(n)$ containing all simple cellular faults defined by the FP and any fault location i with $i \leq n$. Therefore the following notions are meaningful:

- testability of an FP at C_i in $C(n)$,
- redundancy of an FP at C_i in $C(n)$
- full (partial) testability of an FP in $C(n)$,
- full (partial) testability of an FP,
- complete test set for an FP in $C(n)$,
- test complexity of an FP,

- C-testability of an FP.

We finish this section with a lemma which summarizes the correspondence between testability properties of FPs and ILAs. The proof of the lemma follows from the definitions given above and the fact that the number of FPs only depends on properties of C and thus is independent of the length n. Further details are omitted for shortness of the paper.

Lemma 1 *Consider a basic cell C, an ILA $C(n)$ and $C(\cdot)$, the set of ILAs over C. Then the following holds:*

i) A cellular fault F at location $i \leq n$ in $C(n)$ is testable, if and only if there exists an FP that is a component of F and testable at C_i.

ii) $C(n)$ is fully testable, if and only if each FP is fully testable in $C(n)$.

iii) $C(\cdot)$ is fully testable, if and only if each FP is fully testable.

iv) A set $\mathcal{T}$ is a complete test set for $C(n)$ with respect to CFM, if and only if $\mathcal{T}$ is a complete test for each FP in $C(n)$.

v) $C(\cdot)$ has linear test complexity if and only if there exists an FP with linear test complexity.

vi) $C(\cdot)$ is C-testable, if and only if each FP is C-testable.

3 Results

At first we show how to determine the locations where a given FP is redundant. This does not only decide the full (partial) testability of FPs but also determines the redundancies, full (partial) testability of any ILA over C according to Lemma 1. The algorithms merely work on the the state transition graph G_C and the pair graph G_C^2 and therefore have a complexity that is independent of the length of the ILAs.

Consider a yFP $x^{y}_{\hat{y}/*}$ or an xFP $x^y\hat{x}/\hat{x}_f$. Clearly, the state x can be generated at the x-input of C_i in an ILA $C(n)$, if and only if there exists a path in G_C with length $i-1$ and destination x. The existence of such a path is sufficient for the testability of the yFP. If the FP considered is the xFP $x^y\hat{x}/\hat{x}_f$, it is furthermore necessary to distinguish the state $\hat{x}$ from the faulty state $\hat{x}_f$. It is easily seen that the distinguishability of two states $\hat{x}$ and $\hat{x}_f$ at C_i in $C(n)$ is equivalent to the following property of the pair graph G_C^2: A y-distinguishing edge is reachable from the node $(\hat{x}, \hat{x}_f)$ or there exists a path with length $\geq n-i$ that starts in $(\hat{x}, \hat{x}_f)$ and consists only of x-distinguishing edges. Thus, we have

Lemma 2 (Full Testability of FPs) *A yFP $x^y_{\hat{y}/*}$ is fully testable, if and only if there exists a path in G_C that ends in x and contains a cycle.*

An xFP $x^y\hat{x}/\hat{x}_f$ is fully testable, if and only if there exists a path P_1 in G_C and a path P_2 in G_C^2 with the following properties:

> *P_1 ends in x and contains a cycle. P_2 starts in $(\hat{x}, \hat{x}_f)$ and ends with a y-distinguishing edge. If this is not the case, then P_2 consists of x-distinguishing edges, starts in $(\hat{x}, \hat{x}_f)$ and contains a cycle.*

If FPs are not fully testable, then there exist locations at which the FPs are redundant. We completely characterize these locations by using *generation* and *propagation indices*. For an FP $x^y_{\hat{y}/*}$ or $x^y\hat{x}/\hat{x}_f$ define the *generation index* k_g, which gives the locations where the generation of the relevant x-input is possible: if the FP is fully testable then $k_g = \infty$, otherwise k_g is the number of nodes in the longest path in G_C with destination x. Since G_C contains exactly N nodes, it follows easily that $1 \leq k_g \leq N-1$, if $k_g \neq \infty$. The *propagation index* k_p is used to denote the locations (counted from the end of the ILA), where propagation is possible: if the FP is a yFP or fully testable, $k_p = \infty$, otherwise k_p is the number of edges in the longest path in G_C^2, which starts in $(\hat{x}, \hat{x}_f)$ and consists of x-distinguishing edges. Since G_C^2 contains exactly $N(N-1)$ nodes with differing components, it follows that $0 \leq k_p \leq N(N-1)-1$, if $k_p \neq \infty$. ¿From Lemma 2 and the above definitions we directly conclude that the generation and propagation indices of FPs can be efficiently computed by graph algorithms in G_C and G_C^2 and thus are independent of n. If we define as usually $n - \infty$ by $-\infty$, we get the following theorem which characterizes the importance of k_g, k_p.

Theorem 1 (Testability Locations of FPs) *Consider an ILA of length n over C and an FP with generation index k_g and propagation index k_p. Then the FP is testable exactly at the locations in the interval*

$$[n - k_p, k_g] \cap [1, n]$$

k_g (k_p) is independent of n and can be computed in time $O(NM)$ ($O(N^2M)$).

It follows immediately from the theorem and the properties of k_g, k_p given above that partially testable FPs can at most be testable at the first $N-1$ or last $N(N-1)-1$ cells of a given ILA. We therefore obtain

Corollary 1 *Partially testable FPs are C-testable.*

The corollary shows that partially testable FPs are optimally testable, the test complexity of an ILA is therefore determined by the test complexities of the fully testable FPs. We show in the following that the set of FPs splits into exactly two complexity classes and give a characterization of C-testability that allows the construction of a test. The proof of both results contains the application of a lemma which in similar form was first used by Friedman to characterize the C-testability of

ILAs without southern outputs. Because of its importance we restate the lemma and give a short proof.

Lemma 3 (Friedman [1]) *Let T be a test for an xFP $x^y\hat{x}/\hat{x}_f$ in an ILA $C(n)$ with the following properties:*

> *T simultaneously tests the xFP at l different cells C_{i_j} with $l > 2^{N-1}$ and $1 \leq i_1 < i_2 < \ldots < i_l \leq n$. A difference $\hat{x}/\hat{x}_f$ at the x-output lines of a cell C_{i_j} is propagated to the x-output lines of C_{i_l}.*

Then there exists $R \in Y^*$ such that yR repeats x at the x-input lines and $(Ry)^i$ for any $i \in \mathbf{N}$ propagates $\hat{x}/\hat{x}_f$ in horizontal direction, i.e.

$$\begin{aligned} g^X(x, yR) &= x \\ g^X(\hat{x}, (Ry)^i) &\neq g^X(\hat{x}_f, (Ry)^i) \text{ for all } i \in \mathbf{N} \end{aligned}$$

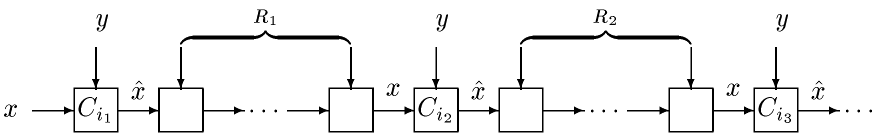

Figure 3: Simultaneous test at l different cells C_{i_j}

Proof: We may assume without loss of generality that $1 = i_1 < i_2 < \ldots < i_l = n$, and T is given by $T = (x, yR_1yR_2\ldots yR_{l-1}y)$ with $R_j \in Y^*$ $|R_j| = i_{j+1} - i_j - 1$ for $j = 1, \ldots, l-1$. For an illustration see Figure 3. The strings R_jy must have the following propagation properties. R_1y has to propagate the difference $\hat{x}/\hat{x}_f$ (test of the xFP at C_{i_1}), i.e.

$$\hat{x} = g^X(\hat{x}, R_1y) \neq g^X(\hat{x}_f, R_1y) =: \hat{r}$$

R_2y has to propagate the differences $\hat{x}/\hat{x}_f$ (test of the xFP at C_{i_2}) and $\hat{x}/\hat{r}$ (test of the xFP at C_{i_1}), i.e.

$$\begin{aligned} \hat{x} = g^X(\hat{x}, R_2y) &\neq g^X(\hat{x}_f, R_2y) \\ \hat{x} = g^X(\hat{x}, R_2y) &\neq g^X(\hat{r}, R_2y) \end{aligned}$$

Generally we have for $j = 1, \ldots, l-1$: $\exists\, X_j \subset X - \{\hat{x}\}$ with: R_jy has to propagate the differences $\hat{x}/\hat{r}\ \forall \hat{r} \in X_j$.

Remember that l is greater than 2^{N-1}, the number of subsets of $X - \{\hat{x}\}$. Since the $X_j \neq \emptyset$ for all j, we conclude the existence of $r, s \in \{1, \ldots, l-1\}$ with $r < s$ and $X_r = X_s$. $R := R_ryR_{r+1}y\ldots R_{s-1}$ then has the desired properties. $\square$

We are now ready to prove that only two test complexity classes for FPs exist.

Theorem 2 (Test Complexity of an FP) *Let $F(\cdot)$ be the set of faults defined by an FP. Then*

$$TC_F(n) = \begin{cases} \Theta(1) & : \quad \textit{if the FP is C-testable} \\ \Theta(n) & : \quad \textit{otherwise} \end{cases}$$

Proof: Consider an FP that is not C-testable. From Corollary 1 we know that this FP must be fully testable. For the proof it suffices to show that a complete test for this FP requires a linear number of tests.

Case 1: The FP is a yFP $x^y_{\hat{y}/*}$.

We show that the yFP requires n tests in an ILA $C(n)$ for all $n \in \mathbf{N}$. Assume to the contrary that there exists $n \in \mathbf{N}$ and the yFP is tested simultaneously at two different locations i and j ($j > i$) in $C(n)$. Then there exists an $R \in Y^{(j-i-1)}$, that generates x from $g^X(x, y)$, i.e. an R with $g^X(x, yR) = x$. R is independent of n, it corresponds to the label of a path in G_C. Use the method already presented in [1] to define a C-test for the yFP as follows:

> Consider the y-input sequence $yRyR\ldots$ and combine it with the x-input x. Then the yFP is tested at the cells $C_1, C_{1+(|R|+1)}, C_{1+2(|R|+1)}, \ldots, C_{1+k(|R|+1)}, \ldots$ in an ILA of arbitrary length. Now consider the $|R|+1$ "translations" of $yRyR\ldots$ defined by $R'yRyR\ldots$ with R' is any suffix of yR with length ≥ 1. Combine these translations with suitable choices of the primary x-input to obtain a C-test for the yFP with $|R|+1$ tests. This is a contradiction to our assumption.

Case 2: The FP is an xFP $x^y\hat{x}/\hat{x}_f$.

If the xFP is not testable at more than one location simultaneously (in any $C(n)$), we know that the xFP requires n tests in $C(n)$. Therefore assume that there exists $C(n)$ and the xFP can be tested with a primary input combination T at $l > 1$ locations $i_1, i_2, \ldots, i_l$ simultaneously. With the same arguments as in Case 1 it follows that the difference $\hat{x}/\hat{x}_f$ at the x-output lines of any cell C_j with $j \in \{i_1, i_2, \ldots, i_{l-1}\}$ cannot be propagated to the y-output lines of a cell C_k with $j < k \leq i_l$, but must be propagated to the x-output lines of C_{i_l}. If this is the case we conclude that $l \leq 2^{N-1}$ as follows: If $l > 2^{N-1}$ Lemma 3 is applicable and yields the existence of a sequence $yRyR\ldots$, that can be combined with the x-input x to test the xFP at the cells $C_1, C_{1+(|R|+1)}, C_{1+2(|R|+1)}, \ldots, C_{1+k(|R|+1)}, \ldots$. As in Case 1 this test and its $|R|$ "translations" define a C-test for the xFP in contradiction to our assumption. Therefore the xFP can be tested at most at 2^{N-1} locations simultaneously, which shows that the test complexity of the xFP is $\geq n/2^{N-1}$ (with N independent of n). $\square$

Lemma 1 and Theorem 2 directly lead to

Corollary 2 (Test Complexity of $C(\cdot)$)

$$TC_C(n) = \begin{cases} \Theta(1) & : \quad \textit{if } C(\cdot) \textit{ is C-testable} \\ \Theta(n) & : \quad \textit{otherwise} \end{cases}$$

In the following we give an algorithm to decide the C-testability of an FP and thereby construct a C-test for this FP if possible. Our criterion for C-testability is a generalization of Friedman's result ([1]) to ILAs that possibly have primary outputs in the south and are not necessarily fully testable.

Theorem 3 (C-testability of an FP) *An FP is C-testable if and only if one of the following cases holds*

i) The FP is partially testable, i.e. the generation index k_g or the propagation index k_p is unequal ∞.

ii) The FP is a yFP $x^{y}_{\hat{y}/*}$ and there exists $R \in Y^*$ with

$$g^X(x, yR) = x$$

iii) The FP is an xFP $x^y \hat{x}/\hat{x}_f$ and there exists $R \in Y^*$ with

$$\begin{aligned} g^X(x, yR) &= x \\ g^Y_{(1,|R|+1)}(\hat{x}, Ry) &\neq g^Y_{(1,|R|+1)}(\hat{x}_f, Ry) \end{aligned}$$

iv) The FP is an xFP $x^y \hat{x}/\hat{x}_f$ and there exists $R \in Y^*$ with

$$\begin{aligned} g^X(x, yR) &= x \\ g^X(\hat{x}, (Ry)^i) &\neq g^X(\hat{x}_f, (Ry)^i) \text{ for all } i \in \mathbf{N} \end{aligned}$$

Proof: The sufficiency of the conditions given in i) to iv) follows from Corollary 1 (Case i)) and by using the sequence $(x, yRyR\ldots)$ and its translations as tests (remaining cases).

For the proof of the necessity we may assume that the C-testable FP considered is fully testable in any ILA $C(n)$ with k tests ($k \in \mathbf{N}$ fixed). If the FP is a yFP $x^{y}_{\hat{y}/*}$, there must exist an $R \in Y^*$ with $g^X(x, yR) = x$, otherwise n tests would be necessary for a complete test of the yFP. This gives the proof for ii). Now assume that we consider an xFP $x^y\ \hat{x}/\hat{x}_f$ and Case iii) does not apply. Then a C-test can only be obtained by propagation in the horizontal direction. Choose n "large enough" with

$$n/k > 2^{N-1}$$

Since the xFP is C-testable with k tests there exists a test which tests the xFP in $C(n)$ at least at $n/k > 2^{N-1}$ locations. Apply Lemma 3 to obtain the existence of R with the desired properties. □

We call a sequence R that fulfills the properties ii) ,iii) or iv) for a fixed FP a *repetition sequence* of the FP. As mentioned before the repetition sequence can be used to specify a C-test of size $|R| + 1$, the *repetitive C-test* with respect to R. A repetitive C-test is minimal, iff the repetition sequence has minimal length. The question how to construct small repetition sequences, if they exist at all, is discussed next. We distinguish the four cases of Theorem 3 and a fifth case that applies if the FP is not C-testable and demonstrate how small (asymptotically optimal) complete test sets can be efficiently constructed.

Case i)

As mentioned before the indices k_g and k_p can be computed by graph algorithms linear in the size of G_C and G_C^2. This also gives a complete test set with $\leq N(N-1)$ tests.

Case ii)

The reachability of x from the node $g^X(x,y)$ in G_C has to be checked. This can be done in time linear in the size of G_C and in the positive case leads to R with $|R| \leq N-1$, since we can omit cycles.

Case iii)

For an xFP $x^y\, \hat{x}/\hat{x}_f$ we at first check, if Case iii) holds. To do so, search for a path P in G_C^2 with the following properties: P starts in node $(\hat{x}, \hat{x}_f)$ and leads to a node $(\hat{x}, \hat{x}')$, the last edge of the path starts in (x, x') and is labeled with y-input y. P must contain at least one y-distinguishing edge. The search can be done in time that is linear in the size of G_C^2: At first, label the nodes in G_C from which x is reachable, this takes time $O(G_C)$. Then start a depth first search from $(\hat{x}, \hat{x}_f)$ and for each y-distinguishing edge e visited decide (in constant time), if x is reachable from the first component of the endpoint of e. In the positive case this algorithm results in an R of size $\leq N(N-1) + N - 1$.

Case iv)

If Case i) and iii) do not apply for an xFP, we have to check the existence of an R with the general properties given in part iv). Algorithms for this problem are considered in [1]. Their worst case runtime is exponential in N and also the length of the resulting R may be exponential in N. In the following we demonstrate that C-testability can be decided efficiently (polynomial in N) also in this general case. If C-testability is fulfilled we thereby construct a "short" sequence R with the properties given in iv).

We start with a modification of the necessary and sufficient condition for C-testability.

Lemma 4 *Consider a fully testable xFP $x^y \hat{x}/\hat{x}_f$ and assume that Case iii) does not apply, i.e. the xFP is not C-testable by propagation to the y-outputs. Then the FP is C-testable, if and only if the following properties hold:*

There exist $x' \in X - \{x\}$ and $R' \in Y^*$ such that yR' repeats x at the x-input lines and R' drives the faulty state $\hat{x}_f$ and the state $g^X(x', y)$ as well to state x', i.e.

$$\begin{aligned} g^X(x, yR') &= x \\ g^X(\hat{x}_f, R') &= x' \\ g^X(x', yR') &= x' \end{aligned}$$

Proof: The sufficiency of the properties given in the lemma follows easily from Theorem 3 iv). We now prove the necessity.

Assume that the xFP $x^y\hat{x}/\hat{x}_f$ is C-testable. Then we know, that there exists an R with the general properties given in part iv) of Theorem 3. If there exists $t \geq 1$ with $g^X(\hat{x}/\hat{x}_f, (Ry)^t) = \hat{x}/\hat{x}_f$, then we easily obtain the correctness of the lemma by setting $R' = (Ry)^{t-1}R$ and $x' := g^X(\hat{x}_f, R')$.

In general, the iterative application of Ry results in a situation depicted in Figure 4.

$$\begin{tikzcd}
\hat{x}/\hat{x}_f \arrow[r, "Ry"] & \hat{x}/\hat{x}_1 \arrow[r, "Ry"] & \hat{x}/\hat{x}_2 \arrow[r, "Ry"] & \ldots \arrow[r, "Ry"] & \hat{x}/\hat{x}_i \arrow[r, "Ry"] & \ldots \arrow[r, "Ry"] & \hat{x}/\hat{x}_{t-1} \arrow[ll, bend left, "Ry"]
\end{tikzcd}$$

Figure 4: Iterative application of Ry

There exists $1 \leq t \leq N-1$ and states $\hat{x}_j$ with the properties: $\hat{x}_j \neq \hat{x}, \hat{x}_f$ for $1 \leq j \leq t-1$, $\hat{x}_l \neq \hat{x}_k$ for $k \neq l, 1 \leq k, l \leq t-1$ and $g^X(\hat{x}_{t-1}, Ry) = \hat{x}_i$ for i with $1 \leq i \leq t-1$. Now choose r with $r(t-i) - 1 \geq i$ and set $r' := r(t-i) - 1$. Then define

$$\begin{aligned}
R' &:= (Ry)^{r'}R \\
x' &:= g^X(\hat{x}_f, R')
\end{aligned}$$

We show that R' and x' fulfill the properties of the lemma. The first and second equation follow directly from the definitions of x', R' and the properties of R. Since $(Ry)^j$ propagates the faulty difference for all $j \in \mathbf{N}$, we also know that $x' \neq x$. It remains to prove the property $g^X(x', yR') = x'$. For this notice, that $g^X(\hat{x}_f, (Ry)^i) = \hat{x}_i$. Since R' repeats R more than i times, we have $x' := g^X(\hat{x}_f, R') = g^X(\hat{x}_f, (Ry)^i(Ry)^{r'-i}R) = g^X(\hat{x}_i, (Ry)^{r'-i}R)$. Furthermore, it holds $g^X(\hat{x}_i, (Ry)^{(t-i)}) = \hat{x}_i$ and thus $g^X(\hat{x}_i, (Ry)^{r'+1}) = g^X(\hat{x}_i, (Ry)^{r(t-i)}) = \hat{x}_i$. Using this properties we obtain

$$\begin{aligned}
g^X(x', yR') &= g^X(g^X(\hat{x}_i, (Ry)^{r'-i}R), yR') \\
&= g^X(\hat{x}_i, (Ry)^{r'-i+1}R') \\
&= g^X(\hat{x}_i, (Ry)^{r'-i+1+r'}R) \\
&= g^X(\hat{x}_i, (Ry)^{(r'+1)+(r'-i)}R) \\
&= x'
\end{aligned}$$

□

The properties of Lemma 4 can be checked by investigation of G_C^3: The xFP $x^y\hat{x}/\hat{x}_f$ is C-testable, iff there exists $x' \in X$ and a path P in G_C^3 from node $(\hat{x}, \hat{x}_f, g^X(x', y))$ to node (x, x', x'). The number of nodes on P can be bounded by $N(N-1)^2$. An upper bound on the time complexity is given by N linear time depth first searches on G_C^3.

Notice, that in a preprocessing step we can easily check the special case mentioned in the proof of Lemma 4: $\exists\, R' \in Y^*$ with $g^X(\hat{x}, R') = x$ and $g^X(\hat{x}/\hat{x}_f, R'y) = \hat{x}/\hat{x}_f$. This takes linear time in the size of G_C^2 and in the positive case the length of R' can be bounded by $N(N-1)-1$.

In all of the preceding cases we thus obtain a complete test, whose size and construction time as well is polynomial in N and M. We now come to the last case that has to be considered.

Case v)

The FP is fully testable but not C-testable. Then compute paths P_1 and P_2 with the properties given in Lemma 2. The computation takes linear time in the size of the graphs considered. P_1, P_2 contain all the information necessary for the generation of a complete test set of the FP in any $C(n)$. So, although the test size is n, a specification of the test can be constructed in time independent of n.

We summarize our bounds for test complexity and computation time in the following

Theorem 4 (Construction and size of FP tests) *Consider a cell C with N = number of x-inputs and M = number of y-inputs. Then the following upper bounds hold for any ILA of size n over C:*

FP belongs to	*size of complete test*	*time complexity of test construction*
Case i	$N(N-1)$	$O(N^2M)$
Case ii	N	$O(NM)$
Case iii	N^2	$O(N^2M)$
Case iv	$N(N-1)^2$	$O(N^4M)$
Case v	n	$O(N^2M)$

i.e. a C-testable FP can be completely tested in any ILA over C with at most $O(N^3)$ test patterns. If the FP is not C-testable a complete test requires n test patterns in an ILA of size n. The construction of the complete test can be performed in time $O(N^4M)$.

We finish this section with a discussion of the question, whether the repetitive C-tests constructed above for a fixed FP have minimal size. It turns out, that, although our construction yields small upper bounds, the computation of the exact bound leads to an *NP*-hard problem with respect to N and M.

Theorem 5 *Consider a cell C given by an $N \times M$ function-table (N = number of x-inputs, M = number of y-inputs) and a C-testable FP $x^y\hat{x}/\hat{x}_f$. Then the construction of a repetitive test set for FP of minimal size is NP-hard.*

	0	1	y	
x	x_∞	x_∞	$\hat{x} = x_1$	x requires y
x_∞	x_∞	x_∞	x_∞	trap-state
	for $j = 1, \ldots m-1$:			
x_j	x_{j+1}	x_{j+1}	x_∞	R of length m
x_m	x	x	x_∞	drives $\hat{x}$ back to x
	for $i = 1, \ldots k, j = 1, \ldots m$:			
$x_{i,j}$	$x_{i,j+1}$	$x_{i,j+1}$	$x_{-\infty}$	$z_j, \bar{z}_j \notin Z_i$
$x_{i,j}$	$x_{i,j+1}$	$x^S_{i,j+1}$	$x_{-\infty}$	$z_j \in Z_i$
$x_{i,j}$	$x^S_{i,j+1}$	$x_{i,j+1}$	$x_{-\infty}$	$\bar{z}_j \in Z_i$
$x^S_{i,j}$	$x^S_{i,j+1}$	$x^S_{i,j+1}$	$x_{-\infty}$	Z_i already satisfied
$x_{i,m+1}$	$x_{-\infty}$	$x_{-\infty}$	$\hat{x}$	clause Z_i is not satisfied
	for $i = 1, \ldots k-1$:			
$x^S_{i,m+1}$	$x_{-\infty}$	$x_{-\infty}$	$x_{i+1,1}$	check of clause Z_{i+1} is prepared
$x^S_{k,m+1}$	$x_{-\infty}$	$x_{-\infty}$	$x_{1,1}$	all clauses satisfied
$x_{-\infty}$	$x_{-\infty}$	$x_{-\infty}$	$\hat{x}$	choices that do not propagate the fault

Table 1: Function table for reduction of 3SAT

Proof:
We give a reduction from 3SAT: Consider a set Z of k clauses over m Boolean variables $z_1, z_2, \ldots, z_m$, each clause consisting of exactly 3 literals. For the proof it is sufficient to construct a function table for a basic cell (without y-outputs), that fulfills the following properties:

1. $\hat{x} = g^X(x, y)$
2. $g^X(x, yR) = x$, iff $|R|$ is a multiple of m.
3. $x^y \hat{x} / \hat{x}_f$ is fully testable and C-testable.
4. $\exists R \in Y^m$ with: $g^X(\hat{x}, R) = x$ and $g^X(\hat{x}, (Ry)^i) \neq g^X(\hat{x}_f, (Ry)^i)$ for all $i \in \mathbf{N}$, iff Z is satisfiable.

We first define state transitions that fulfill the properties 1 and 2. For this consider the state x, m states $x_1, x_2, \ldots, x_m$ and a "trap"-state x_∞. As y-inputs we use 3 values $0, 1, y$. Now set

$$\hat{x} := x_1$$

and consider the state transitions given for these states in the first part of Table 1. Properties 1 and 2 from above follow directly.

To fulfill properties 3 and 4 we proceed as follows. For each clause Z_i $(i = 1, \ldots, k)$ define $2m + 2$ states $x_{i,j}, x^S_{i,j}$ $(j = 1, \ldots, m+1)$. These states are used to check, if clause Z_i is "satisfied" by a given choice of 0's and 1's. Furthermore, we have a second "trap"-states $x_{-\infty}$. As y-inputs the 3 values $0, 1, y$ are used again. Set

$$\hat{x}_f := x_{1,1}$$

Consider the state transitions for $x_{i,j}, x^S_{i,j}$ and $x_{-\infty}$ defined in the second part of Table 1. Using these definitions we can now finish the proof.

It is easy to see that the FP $x^y\hat{x}/\hat{x}_f$ is fully testable and C-testable as well: For $i = 1, \ldots k$ choose $R_i \in \{0,1\}^m$ that satisfies clause Z_i. Then $R_1yR_2y\ldots R_ky$ defines a C-test. On the other hand, since properties 1 and 2 are fulfilled, a sequence of length at least m is required to drive $\hat{x}$ back to x and thus we get that each test C-test has at least cardinality $m+1$.

It remains to prove that property 4 holds. If Z is satisfiable, there exists a satisfying assignment $R \in \{0,1\}^m$, which drives $\hat{x}$ back to x, i.e. $g^X(\hat{x}, R) = x$ and fulfills all clauses, i.e. $g^X(\hat{x}, (Ry)^i) \neq g^X(\hat{x}_f, (Ry)^i)$ for all $i \in \mathbf{N}$. Vice versa, if there exists R of length m with $g^X(\hat{x}, R) = x$ and $g^X(\hat{x}, (Ry)^i) \neq g^X(\hat{x}_f, (Ry)^i)$ for all i, it follows that the first k repetitions of R verify the satisfiability of $Z_1, Z_2, \ldots, Z_k$ successively. $\square$

4 Conclusions and Open Problems

We have characterized the testability of one-dimensional, unilateral iterative logic arrays (ILAs) with respect to the cellular fault model. ILAs considered in this paper consist of horizontally connected copies of a basic cell C. C itself is always required to be combinational, it may have vertical output lines. Contrary to the classical model we did not require the full testability of the circuits considered but allowed redundancies. Fault patterns (FPs) were introduced to handle redundant and testable faults as well. It turned out that redundancies can be completely characterized and efficiently computed. (This information can be used to simplify basic cells at redundant locations and thus obtain an inhomogeneous ILA with removed redundancies and smaller area requirements. However the modification may change the test complexity of the ILA!)

Based on the analysis of redundancies we showed that the test complexity of partially testable FPs is $O(1)$. A characterization of C-testability obtained by Friedman in [1] for the case that C has no vertical outputs was then used together with a detailed analysis of the remaining cases to prove the following:

A fully testable FP is either C-testable or linear testable in the length of the ILA. ILAs therefore split into exactly two complexity classes: ILAs with optimal (= constant) test complexity and ILAs with worst (= linear) test complexity. We further gave new necessary and sufficient conditions for C-testability of an FP which allow the computation of (asymptotically) optimal test sets in all cases. Upper time bounds for the algorithms are also provided. Since the complexity of the algorithms is independent of the size of the ILA considered, complexity is measured relatively to the size of the function table for the basic cell. We were able to prove that all C-tests can be chosen to have small polynomial size with the help of polynomial time algorithms. We further showed that the computation of an optimal repetitive C-test cannot be performed efficiently.

All in all, the results obtained in this paper demonstrate that the restriction to interesting classes of Boolean circuits may lead to both, a complete characterization from the point of "test complexity theory" and practicable test generation algorithms.

Nevertheless at the end we should mention some questions which are left open in this paper. The test complexity classes for other types of ILAs, e.g. sequential and/or two-dimensional ones, are not characterized until now. It is only known at the moment, that for two-dimensional ILAs a gap theorem does not hold: there exist C-testable, $\sqrt{n}$-testable, n-testable ILAs (n = size of the ILA) and even ILAs, for which the testability problem is undecidable.

References

[1] A.D. Friedman. Easily testable iterative systems. *IEEE Trans. on Comp.*, C-22:1061–1064, December 1973.

[2] S.H. Unger. Pattern recognition using two dimensional, bilateral, iterative, combination, switching circuits. In *Proc. Symp. Math. Theory of Automata*, pages 577–591, April 1962.

[3] T. Sridhar and J.P. Hayes. Design of easily testable bit-sliced systems. In *IEEE Trans. on Comp.*, pages 563–571, August 1981.

[4] T.W. Ku and M. Soma. Minimal overhead modification of iterative logic arrays for c-testability. In *Proc. IEEE Int. Test Conf.*, pages 964–969, 1990.

[5] F.C. Hennie. *Iterative Arrays of Logical Circuits.* M.I.T. Press, Cambridge, Mass., 1961.

[6] W.H. Kautz. Testing for faults in combinational cellular logic arrays. In *Proc. 5th Annual Symp. on Switching and Automata Theory*, pages 161–1744, 1967.

[7] C.D. Weiss. Optimal synthesis of arbitrary switching functions with regular arrays of two-input one-output switching elements. *IEEE Trans. on Comp.*, C-18:839–856, September 1969.

[8] M. Cappa and V.C. Hamacher. An augmented iterative array for high speed binary division. *IEEE Trans. on Comp.*, C-22:172–176, 1973.

[9] A. Vergis and K. Steiglitz. Testability conditions for bilateral arrays of combinational cells. *IEEE Trans. on Comp.*, C-35(1):13–22, January 1986.

[10] C.W. Wu and P.R. Capello. Easily testable iterative logic arrays. *IEEE Trans. on Comp.*, C-39(5):640–652, May 1990.

[11] M. Abramovici and M.A. Breuer and A.D. Friedman. *Digital Systems Testing and Testable Design.* Computer Science Press, 1990.

[12] J. Galiay, Y. Crouzet, and M. Vergniault. Physical versus logical fault models MOS LSI circuits: impact on their testability. *IEEE Trans. on Comp.*, C-29(6):527–531, June 1980.

[13] C.C. Beh, K.H. Aray, C.E. Radke, and K.E. Torku. Do stuck-at fault models reflect manufacturing defects? In *Proc. IEEE Int. Test Conf.*, pages 35–42, 1982.

[14] P. Banerjee and J.A. Abraham. Generating tests for physical failures in MOS logic circuits. In *Proc. IEEE Int. Test Conf.*, pages 554–559, 1983.

[15] P. Lamoureux and V.K. Agarwal. Non-stuck-at fault detection in NMOS circuits by region analysis. In *Proc. IEEE Int. Test Conf.*, pages 129–137, 1983.

[16] W. Maly. Realistic fault modeling for VLSI testing. In *Proc. 24th Design Automation Conf.*, pages 173–180, 1987.

[17] F.J. Ferguson and J.P. Chen. Extraction and simulation of realistic CMOS faults using inductive fault analysis. In *Proc. IEEE Int. Test Conference*, pages 475–484, 1988.

[18] R.L. Wadsack. Fault modeling and logic simulation of CMOS and MOS integrated circuits. *The Bell System Technical Journal*, 57(5), 1978.

[19] J.A. Waicukauski, E. Lindbloom, B. Rosen, and V. Iyengar. Transition fault simulation by parallel pattern single fault propagation. In *Proc. IEEE Int. Test Conf.*, pages 542–549, September 1986.

[20] G.L. Smith. A model for delay faults based upon paths. In *Proc. of IEEE Int. Test Conf.*, pages 342–349, September 1985.

[21] B. Becker and U. Sparmann. A uniform test approach for RCC-Adders. In *Proc. of the 3rd Aegean Workshop on Parallel Computation and VLSI Theory, Lecture Notes in Comp. Sci. 319*, pages 288–300, 1988.

[22] B. Milne. Testability, 1985 technology forecast. *Electronic Design*, 10:143–166, 1985.

[23] T.W. Williams, W. Daehn, M. Gr"utzner, and C.W. Starke. Comparison of aliasing errors for primitive and non-primitive polynomials. In *Proc. IEEE Int. Test Conf.*, pages 282–288, 1986.

[24] M. S. Abadir and H. K. Reghbati. Functional testing of semiconductor random access memories. *Computing Surveys*, 15:175–198, 1983.

[25] W. Daehn and J. Mucha. A hardware approach to self-testing of large PLA's. *IEEE Trans. on Comp.*, C-30:829–833, 1981.

[26] H. Fujiwara and K. Kinoshita. A design of programmable logic arrays with universal tests. *IEEE Trans. on Comp.*, C-30(11):823–828, 1981.

[27] J. Ferguson and J.P. Chen. The design of two easily-testable VLSI array multipliers. In *Proc. of the 6th Symp. on Computer Arithmetic*, pages 2–9, June 1983.

[28] B. Becker and J. Hartmann. Optimal-time multipliers and c-testability. In *Symp. on Parallel Algorithms and Architectures (SPAA'90)*, pages 174–179, 1990.

[29] B. Becker and U. Sparmann. Computations over finite monoids and their test complexity. In *19th Int. Symp. on Fault-Tolerant Computing*, pages 299–306, 1989.

[30] W.T. Cheng and J. H. Patel. Testing in two-dimensional iterative logic arrays. In *Proc. of the 16th Int. Symp. on Fault Tolerant Computing Systems*, pages 76–81, July 1986.

[31] R. Parthasarathy and S.M. Reddy. A testable design of iterative logic arrays. *IEEE Trans. on Comp.*, C-30:833–841, 1981.