Publ. Math. Debrecen
48 / 3-4 (1996), 201–215

Applications of automata and languages to testing[1]

By J. A. BRZOZOWSKI (Waterloo) and H. JÜRGENSEN (London)

Abstract. In this paper we survey applications of automaton and language theory to the problems of diagnosis and testing of sequential circuits. We discuss a formal framework based on Mealy automata for defining faults, for modelling the process of diagnosis and detection, for describing the generation of test sequences, for representing multiple faults as composite faults, for the complexity analysis of testing and diagnosis, and for the evaluation of randomized testing methods. Several applications to faults in random access memories are included.

1. Circuit Diagnosis and Testing

While it may be sometimes possible to prove the correctness of a program, it is never possible to prove the correctness of a circuit. Consequently, testing is an indispensable step in the production of VLSI circuits. Present-day digital circuit technology involves extremely sophisticated processes, and typical circuits contain hundreds of thousands of components on a single chip. It is not surprising then that test generation, test application, and the evaluation of test results is time-consuming and expensive.

The *testing problem* is as follows: Given a circuit under test, determine whether it is correct or faulty. The general *diagnosis problem* is to obtain even more detailed information, namely, when the circuit is not correct, which fault is present. Thus the term 'diagnosis' includes 'testing.'

The typical diagnosis set-up is shown in Figure 1.1. The circuit under test (CUT) receives test sequences from a test generator (TG). By

Mathematics Subject Classification: 94C12, 68Q68.
[1]This work was supported by the Information Technology Research Centre of Ontario and by the Natural Sciences and Engineering Research Council of Canada under Grants OGP0000871 and OGP0000243.

observing the output of the circuit under test, one tries to draw conclusions about the circuit. Thus, as an observer, one knows both the input sequence and the output response. Furthermore, we assume that one also knows the intended 'good' circuit and the fault model, that is, the set of possible faulty circuits.

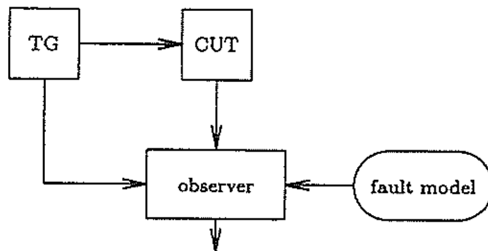

Figure 1.1. Diagnosis set-up.

In this paper we show how automaton and language theoretic methods can be applied to various problems dealing with testing and diagnosis of sequential circuits. In Section 2, we formally define the fault model in terms of Mealy automata. In Section 3, we describe several fault types occurring in random-access memories (RAMs); RAMs are used here as a source of convenient and interesting examples that illustrate the theory discussed in this paper. Section 4 introduces a general framework for diagnosis and testing of sequential machines. The languages defined by sets of test and diagnosis sequences are treated in Section 5. The composition of single faults leads to an algebra of faults described in Section 6. Section 7 illustrates one proof technique for deriving lower bounds on the length of test sequences. It also includes a summary of the known lower bound results. Section 8 discusses probabilistic testing techniques.

2. Fault Model

A defect introduced during the manufacture of a circuit may lead to a change in the circuit's functional behaviour. It is impossible to predict all the physical defects that are likely to occur. Many years of experience have shown that high-level functional fault models suffice to describe the effects of most of the likely physical defects.

In this paper, we focus on problems concerning the diagnosis and testing of sequential circuits. We describe both the intended behaviour of the circuit and the faults in the functional fault model by finite Mealy automata. A similar approach was already used in 1964 by Poage and McCluskey [15] and by Hennie [11], [12]. The *good machine type* is a Mealy automaton $A^0 = (Q^0, X, Y^0, \delta^0, \lambda^0)$ where Q^0 is the set of states,

X is the input alphabet, Y^0 is the output alphabet, $\delta : Q^0 \times X \to Q^0$ is the transition function, and $\lambda : Q^0 \times X \to Y^0$ is the output function. A *fault type* is another Mealy automaton $A^i = (Q^i, X, Y^i, \delta^i, \lambda^i)$ which may differ from A^0 in the set of states, the output alphabet, the transition function, and the output function. A *good machine* (A^0, q^0) is the good machine type A^0 initialized to some state $q^0 \in Q^0$. Similarly, a *fault* is a fault type initialized to some state. A *fault model* $\mathcal{F}$ for A^0 is defined by the following items: A finite family of fault types $A^1, \ldots, A^m$ and, for $i = 0, 1, \ldots, m$, a set $P^i \subseteq Q^i$ of *potential initial states* of A^i. Given such a fault model, one would compare the circuit under test with every (A^i, q^i) where $i = 0, 1, \ldots, m$ and $q^i \in P^i$. To simplify the mathematical presentation we assume that all state sets Q^i are subsets of a common state set Q and that all output alphabets Y^i are subsets of a common alphabet Y.

3. Memory Faults

Throughout the paper we are using RAMS as convenient examples of sequential circuits. Note, however, that the theory is not limited to RAMs.

A *fault-free n-cell RAM type* is a Mealy automaton

$$M = (Q, X, Y, \delta, \lambda),$$

where $Q = \{0,1\}^n$, $X = \bigcup_{i=1}^n X_i$ with $X_i = \{r^i, w_0^i, w_1^i\}$, $Y = \{0, 1, \$\}$, and δ and λ are defined by

$$\begin{aligned}
\delta((q_1, \ldots, q_i, \ldots, q_n), r^i) &= (q_1, \ldots, q_i, \ldots, q_n), \\
\delta((q_1, \ldots, q_i, \ldots, q_n), w_0^i) &= (q_1, \ldots, 0, \ldots, q_n), \\
\delta((q_1, \ldots, q_i, \ldots, q_n), w_1^i) &= (q_1, \ldots, 1, \ldots, q_n), \\
\lambda((q_1, \ldots, q_i, \ldots, q_n), r^i) &= q_i, \\
\lambda((q_1, \ldots, q_i, \ldots, q_n), w_0^i) &= \$,
\end{aligned}$$

and

$$\lambda((q_1, \ldots, q_i, \ldots, q_n), w_1^i) = \$.$$

The input r^i represents the read operation applied to cell i, and the input w_a^i represents the operation of writing the value of a into cell i. The output symbol $ represents the fact that no output is produced when one writes into a memory cell. Faulty RAM types are represented similarly, as illustrated in the three examples below.

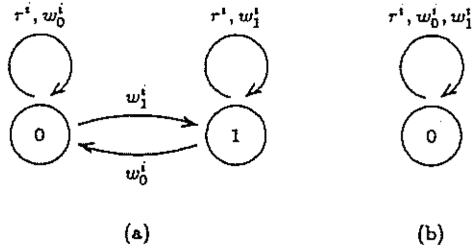

Figure 3.1. The stuck-at fault type $M^{i=0}$: (a) the fault-free RAM type; (b) the stuck-at fault type. Only the states of cell i are shown.

Example 3.1. A RAM with cell i stuck-at-a is a Mealy automaton

$$M^{i=a} = (Q^{i=a}, X, Y, \delta^{i=a}, \lambda^{i=a}),$$

where

$$Q^{i=a} = \{q \mid q = (q_1, \dots, q_{i-1}, a, q_{i+1}, \dots, q_n)\},$$

X and Y are as in the fault-free memory, $\delta^{i=a}$ is the restriction of δ to $Q^{i=a} \times X$, *except* for the transitions under input $w_{\overline{a}}^i$ where $\overline{a}$ is the complement of a. This input fails to write $\overline{a}$ in cell i, that is,

$$\delta^{i=a}((q_1, \dots, q_{i-1}, a, q_{i+1}, \dots, q_n), w_{\overline{a}}^i) = (q_1, \dots, q_{i-1}, a, q_{i+1}, \dots, q_n).$$

Finally, $\lambda^{i=a}$ is the restriction of λ to $Q^{i=a} \times X$. An example of this fault type is given in Figure 3.1, where cell i is stuck-at-0.

Example 3.2. An n-cell RAM type with a *transition* fault in cell i is a Mealy automaton $M^{i\neg a} = (Q, X, Y, \delta^{i\neg a}, \lambda)$, where $\delta^{i\neg a}$ behaves like δ *except* that, when the input w_a^i is applied to any state in which cell i is in state $\overline{a}$, the write operation is unsuccessful, that is,

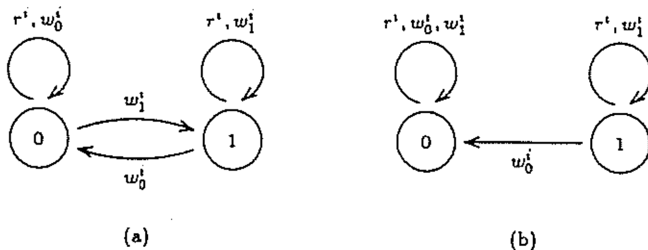

Figure 3.2. The transition fault type $M^{i\neg 1}$: (a) the fault-free RAM type; (b) the transition fault type. Only the states of cell i are shown.

$$\delta^{i\neg a}((q_1, \dots, q_{i-1}, \overline{a}, q_{i+1}, \dots, q_n), w_a^i) = (q_1, \dots, q_{i-1}, \overline{a}, q_{i+1}, \dots, q_n),$$

as in the stuck-at-$\overline{a}$ fault type. Note that the transition-fault type automaton is not strongly connected: Once an $\overline{a}$ is written into cell i, it can never become a again. Figure 3.2 illustrates the distinction between $M^{i=0}$ and $M^{i\neg 1}$. A fault type may represent several different faults depending on the initial state. Thus the fault type $M^{i\neg 1}$ represents two different transition faults. If the initial state is 0, the fault is exactly the same as the corresponding cell i stuck-at-0 fault. But, for the initial state 1, the two faults differ.

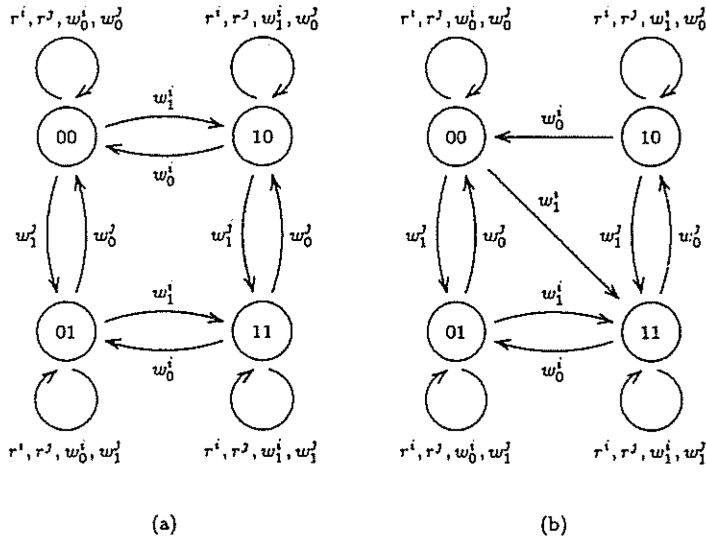

Figure 3.3. The coupling fault type $M^{i1\Rightarrow j1}$: (a) the fault-free RAM type; (b) coupling fault type. Only the states of cells i and j are shown. The first component of a state corresponds to cell i, the second one to cell j.

Example 3.3. A coupling fault type involves two different memory cells, i and j. When the present states of cells i and j are $\overline{a}$ and $\overline{b}$, respectively, and a is written into cell i, then cell j, as well as cell i, changes state. The fault type is denoted $M^{ia\Rightarrow jb} = (Q, X, Y, \delta^{ia\Rightarrow jb}, \lambda)$ where $\delta^{ia\Rightarrow jb}$ behaves like δ *except* for the transitions under input w_a^i when the states of cells i and j are $\overline{a}$ and $\overline{b}$, respectively; under these conditions

$$\delta^{ia\Rightarrow jb}((q_1, \dots, q_{i-1}, \overline{a}, q_{i+1}, \dots, q_{j-1}, \overline{b}, q_{j+1}, \dots, q_n), w_a^i)$$
$$= (q_1, \dots, q_{i-1}, a, q_{i+1}, \dots, q_{j-1}, b, q_{j+1}, \dots, q_n),$$

for the case where $i < j$. The case $i > j$ is similar. Figure 3.3 illustrates the fault type $M^{i1\Rightarrow j1}$.

4. Observer

Let A^0 be the good machine type and let $\mathcal{F}$ be a fault model for A^0 involving the fault types $A^1, \ldots, A^m$. Let A' be the circuit under test. A test generator applies a sequence of inputs to A'. An evaluation unit receives both the test inputs and the outputs of A' and compares them to the behaviours of each of the machines described by the fault model. It then outputs a diagnosis of A' on the basis of the currently available information. This evaluation unit is modelled by the initialized deterministic semi-automaton $\Delta = \Delta(A^0, \mathcal{F}) = (D, X \times Y, \delta, d_0)$, called the *(deterministic) observer,* which is defined as follows.

For $i = 0, 1, \ldots, m$, let $A^i = (Q^i, X, Y^i, \delta^i, \lambda^i)$ and let $\widetilde{Q}^i = Q^i \cup \{\omega\}$ where ω is a new state symbol. Moreover, for $q \in Q^i$ and $(x, y) \in X \times Y^i$, let

$$\widetilde{\delta}^i(q, (x, y)) = \begin{cases} \delta^i(q, x), & \text{if } \lambda^i(q, x) = y, \\ \omega, & \text{if } \lambda^i(q, x) \neq y, \end{cases}$$

and let $\widetilde{\delta}(\omega, (x, y)) = \omega$. A state $d \in D$ is a tuple with components d_q^i for $i = 0, 1, \ldots, m$ and $q \in P^i$. Thus $D = \{d \mid d_q^i \in \widetilde{Q}^i,\ i = 0, 1, \ldots, m\ , q \in P^i\}$. The initial state d_0 of Δ is given by $[d_0]_q^i = q$. For the transition function, $\delta(d, (x, y)) = e$ if and only if $e_q^i = \widetilde{\delta}^i(d_q^i, (x, y))$ for all i and q.

The maximum amount of information that can be obtained from a test of a given circuit A' is

- the machine type, that is, the value of the index i such that $A' = A^i$,
- the initial state q of A', and
- the present state q'.

The set

$$K = \{(i, q, q') \mid i = 0, 1, \ldots, m,\ q \in P^i,\ q' \in Q^i\}$$

describes the set of all possible outcomes of such complete identification. Frequently, one does not need all this information. For example, it may suffice to identify the machine type or just to determine whether A' is faulty or not. The properties we may wish to identify for a given purpose can be conveniently specified by a partition $B = (B_1, \ldots, B_r)$ of the set K, where $B_1, \ldots, B_r$ are the blocks of B. In other words, we are satisfied with knowing the block of the partition to which the circuit under test belongs.

Given such a partition B, we say that a state d of Δ is *B-decided* if the *contents of* d, that is, the set

$$\{(i, q, q') \mid (i, q, q') \in K,\ q' = d_q^i \neq \omega\}$$

is contained in a block of B. An input word w is said to *B-diagnose* if and only if $\delta(d_0,(w,v))$ is B-decided for every output word v with $|v| = |w|$. The following are some typical useful partitions:

- The *maximum information partition* B_{MI} having the singleton sets $\{(i,q,q')\}$ for $i = 0,1,\dots,m$, $q \in P^i$, and $q' \in Q^i$ as blocks. A state d of Δ is B_{MI}-decided if its contents is empty or if there is exactly one triple (i,q,q') such that $d^i_q \neq \omega$ and, moreover, $d^i_q = q'$. In the test situation the latter case means that the behaviour of the circuit A' under test has been consistent with that of A^i started in state q, and that the present state of (A^i,q) is q'.
- The *machine partition* B_{M} with the sets $\{(i,q,q') \mid q' \in Q^i\}$ as blocks for $i = 0,1,\dots,m$ and $q \in P^i$.
- The *machine type and present state partition* B_{TPS} with the sets $\{(i,q,q') \mid q \in P^i\}$ as blocks for $i = 0,1,\dots,m$ and $q' \in Q^i$.
- The *machine type partition* B_{T} with the sets $\{(i,q,q') \mid q \in P^i, q' \in Q^i\}$ as blocks for $i = 0,1,\dots,m$.
- The *fault partition* B_{F} with the two sets $\{(0,q,q') \mid q \in P^0, q' \in Q^0\}$ and $\{(i,q,q') \mid i = 1,\dots,m,\ q \in P^i, q' \in Q^i\}$ as blocks.

The names of these partitions indicate the diagnosis goals defined by the partitions. For instance B_{F} defines the goal of detecting the presence of a fault.

The method of expressing the diagnosis goal in terms of a partition B and the notion of B-decidedness generalize the work of Hennie on *distinguishing* and *homing* sequences [11], [12], [4]. The complexity of determining distinguishing sequences has been studied in [19].

Given a diagnosis goal B, one can reduce the observer in two ways. First, for *diagnosability reduction* one considers all B-decided states of Δ as equivalent and then reduces using standard techniques. Sequences in X^* are then classified as either diagnosing or non-diagnosing. Second, for *diagnosis reduction* one assigns Moore outputs $\gamma(d)$ to the states d of Δ as follows: $\gamma(d)$ is 0, j, or ∞ if the contents of d is not B-decided, non-empty and contained in block B_j of B, or empty, respectively. One then applies classical reduction techniques to this Moore machine to obtain the diagnosis reduction of Δ. Sequences in X^* are now classified as non-diagnosing, diagnosing block B_j, or diagnosing a fault outside the fault model.

5. Diagnosing Languages

Given an observer $\Delta = (D, X \times Y, \delta, d_0)$ and a diagnosis goal B, one

can construct a non-deterministic finite acceptor

$$\overline{\Delta} = (D, X, \overline{\delta}, \{d_0\}, F)$$

where F is the set of B-decided states of Δ and

$$\overline{\delta}(d, x) = \{d' \mid \exists y \in Y : \ \delta(d, (x, y)) = d'\}.$$

Thus a word w B-diagnoses if and only if every computation of $\overline{\Delta}$ on w is an accepting computation. A word w is said to be a *reduced B-diagnosing word* if w B-diagnoses and, if w' results from w by omitting some symbols, then w' does not B-diagnose. Let $L_B(\mathcal{F})$ be the set of B-diagnosing words for the fault model $\mathcal{F}$.

Theorem 5.1 ([4], [13]). *The set $L_B(\mathcal{F})$ is regular. An automaton accepting this set can be algorithmically constructed from the fault model. The set of reduced B-diagnosing words is a hypercode, hence finite, which can be computed from the fault model.*

Fault coverage typically refers to two classes of faults, the fault model $\mathcal{F}_1$ and another fault set $\mathcal{F}_2$. For instance, in the case of memory testing, one could ask the question to what extent testing for stuck-at faults would also test for coupling faults. For a diagnosis goal B, the set $\mathcal{F}_1$ is said to *B-cover* $\mathcal{F}_2$ if $L_B(\mathcal{F}_1) \subseteq L_B(\mathcal{F}_2)$. When $\mathcal{F}_1$ B-covers $\mathcal{F}_2$ any test for faults in $\mathcal{F}_1$ is also a test for the faults in $\mathcal{F}_2$.

Theorem 5.2 ([13], [18]). *B-covering is a decidable pre-order.*

The following property indicates how the set of B-diagnosing words for a set of faults can be constructed from observers for parts of the set.

Theorem 5.3 ([13], [18]). *Let $\mathbf{F}$ be a family of fault models. Then*

$$L_B\left(\bigcup_{\mathcal{F}\in\mathbf{F}} \mathcal{F}\right) = \bigcap_{\mathcal{F}\in\mathbf{F}} L_B(\mathcal{F}).$$

6. Fault Algebra

In the literature on circuit testing one usually starts with the concept of a 'single fault'. By that we mean the effect of a simple defect, such as a RAM cell being 'stuck-at-0'. Frequently, tests are derived so as to detect all such single faults. An important question then arises whether the test so designed will detect 'multiple faults', that is, faulty circuits in which several defects are present at the same time. Perhaps surprisingly, the definition of multiple faults presents several challenging problems.

Suppose we consider a good machine type A^0 and two fault types A^1 and A^2. We would like to construct the fault type $A^{1,2}$ which models the simultaneous presence of the defects represented by A^1 and A^2. One should be able to compute $A^{1,2}$ from A^1 and A^2; of course, this construction will also depend on A^0, but A^0 is fixed. In summary, we are looking for a binary operation $\circ$ such that $A^{1,2} = A^1 \circ A^2$. Obviously, this operation should be associative, commutative and idempotent. In other words, the set of all multiple (and single) faults is necessarily a semilattice. To define the operation $\circ$ completely, one needs to take into account the physical aspects of the defects. The resulting operation on automata has not been considered before in automaton theory.

For most fault models the composition problem is open. The first formal definition of composition of RAM faults was given in [1]. That definition was made in terms of the general class of so-called 'delta' faults, but turned out to be applicable only to coupling faults in RAMs. The deficiency of this first definition was noted in [6], where it was observed that this composition produced undesirable results for transition faults. In [6] a different composition operation was then defined for transition faults, and yet another one for stuck-at faults. In [5] we introduced a single composition operation applicable to all three cases. A similar model was independently developed in [10]. This solves the fault composition problem for the classical fault model of stuck-at, transition, and coupling faults in RAMs [17] completely. In order to achieve this, it turned out to be necessary to introduce a more detailed automaton model—the so-called component automaton model—than that of Mealy automata.

If the two fault types A^1 and A^2 have no states in common they are *incompatible* and their composition is the *empty automaton*, denoted by $\mathbf{0}$, with the empty transition and output functions. Incompatibility represents the fact that the composite fault resulting from the simultaneous presence of two faults is outside our functional fault model. The physical defects represented by two such incompatible faults can be present simultaneously. For example, a cell could have both a short to ground and to the supply voltage. Separately, each of these shorts would behave like a stuck-at fault in the functional fault model. When both are present the cell output may take on an intermediate value between ground and the supply voltage. Thus the binary logic model would no longer apply.

In general, the composition of two fault types results in a new fault type, distinct from the two factors. In some cases, however, two fault types A^1 and A^2 may satisfy the relation $A^1 \circ A^2 = A^1$. This represents the *masking* of A^2 by A^1. The physical causes leading to both types of faults can be present; however, the effects of only one of the two faults are manifested.

The following theorem provides a complete characterization of compatibility and masking in the classical RAM fault model.

Theorem 6.1 ([5]). *The set*

$$\begin{aligned}&\{M^{i=0} \circ M^{i=1} = 0 \mid i = 1, \dots, n\}\\ \cup&\{M^{i=a} \circ M^{i\neg b} = M^{i=a} \mid i = 1, \dots, n,\ a, b \in \{0,1\}\}\\ \cup&\{M^{ia \Rightarrow jb} \circ M^{j=c} = M^{j=c} \mid i, j = 1, \dots, n,\ a, b, c \in \{0,1\}\}\end{aligned}$$

is a minimal set of defining relations for the semilattice generated by stuck-at, transition, and coupling fault types in an n-bit RAM.

7. Lower Bounds

Lower bounds on the length of test sequences for the detection of several types of faults in RAMs have been established using language theoretic tools [1], [6], [7]. In this section we illustrate one of the proof techniques for the class of so-called toggling faults. We also tabulate the known lower bound results.

A *single toggling fault type* is the composition of two coupling fault types $M^{ia \Rightarrow jb}$ and $M^{ia \Rightarrow j\bar{b}}$. We denote this fault type by $M^{ia \Rightarrow j}$. A *toggling fault type* is the composition of any number of single toggling fault types.

We consider a faulty n-bit RAM type M containing only a toggling fault. For the purpose of establishing lower bounds we consider the special case when the good and the faulty RAMs are both initialized to the same state q. Assume now that q is fixed. A test sequence $t \in X^*$ is called *non-repetitious* if every write operation in t results in a state change in the good memory (M^0, q). One verifies that, for any test t that decides between (M^0, q) and the faulty RAM (M, q), there is a non-repetitious test which also decides between (M^0, q) and (M, q) and which has the same length as t.

Any test sequence $t \in X^*$ can be decomposed with respect to operations on a particular cell j as follows:

$$t = u_1 v_1 r^j \cdots u_h v_h r^j t'$$

where $t' \in \left(X \setminus \{r^j\}\right)^*$ and, for $k = 1, \dots, h$, u_k is either empty or is in $\left(X \setminus \{r^j\}\right)^* \cap X^*\{w_0^j, w_1^j\}$ and v_k does not contain any operations to cell j.

If we fix cell j there are $2(n-1)$ possible single toggling fault types to cell j. For each such type $M^{ia \Rightarrow j}$, and for each test sequence t decomposed

as above, we define a binary vector e^{ia} of length h such that e_k^{ia} is the number, modulo 2, of w_a^i operations in v_k. Also, for each single toggling fault type $M^{ia \Rightarrow j}$ we define a binary value

$$\pi^{ia} = \begin{cases} 1, & \text{if } M^{ia \Rightarrow j} \text{ occurs in } M, \\ 0, & \text{otherwise.} \end{cases}$$

For example, let $n = 4$, $j = 4$, and let $t = u_1 v_1 r^4 u_2 v_2 r^4 u_3 v_3 r^4 t'$ where

$$\begin{array}{ll} u_1 \text{ is empty}, & v_1 = r^3 w_1^2 r^1, \\ u_2 = r^2 w_0^3 r^3 w_1^3 r_1 w_1^4, & v_2 = w_1^1 w_0^3 w_0^1 r^3 w_1^3 w_0^2 w_1^2 w_0^2 w_1^1 w_1^2 w_0^2, \\ u_3 = w_0^1 r^1 w_1^4 r^3 w_1^2 w_0^4, & v_3 = w_1^1 w_0^1 w_1^1 w_0^2 w_1^2 w_0^3, \\ t' = w_0^2 r^2. & \end{array}$$

The corresponding vectors are

$$\begin{array}{l} e^{10} = (0,1,1),\ e^{11} = (0,0,0),\ e^{20} = (0,1,1), \\ e^{21} = (1,0,1),\ e^{30} = (0,1,1),\ e^{31} = (0,1,0). \end{array}$$

Suppose now that

$$M = M^{11 \Rightarrow 4} \circ M^{20 \Rightarrow 4} \circ M^{31 \Rightarrow 4}.$$

Then we have

$$\pi^{10} = 0,\ \pi^{11} = 1,\ \pi^{20} = 1,\ \pi^{21} = 0,\ \pi^{30} = 0,\ \pi^{31} = 1.$$

Now let

$$e = \sum_{i \neq j} \sum_{a \in \{0,1\}} \pi^{ia} e^{ia}$$

where the summation is taken modulo 2.

Lemma 7.1 ([1], [6]). *Let M be a toggling fault type to cell j. A non-repetitious test t decides between (M^0, q) and (M, q) if and only if e is not the null vector.*

Continuing with the example above, one finds that $e = (0,0,1)$. Hence, the sequence t in the example decides between the good and faulty machines.

Lemma 7.1 is the key to proving the following lower bound on the length of test sequences for toggling faults.

Theorem 7.1 ([1], [6]). *Any test that detects all toggling faults in an n-bit RAM has at least $2n^2 - 2n$ read operations and at least $3n$ write operations.*

Since tests of length $2n^2 + n$ exist this lower bound is tight [1], [6]. Table 7.1 summarizes known lower bounds and known test lengths for several classes of coupling faults [7]. In general, a coupling or toggling fault is said to be *k-limited* if it is the composition of up to k single coupling or toggling faults.

Table 7.1. Summary of lower bounds and best known test lengths for toggling and coupling faults in n-bit RAMs.

Fault Model	Best Lower Bound	Best Known Test Length	Comments
general toggling	$2n^2 + n$	$2n^2 + n$	optimal for $n \geq 2$.
2-limited toggling	$n\lfloor \log_2(n-1)\rfloor$ $+5n$	$4n\lceil \log_2 n\rceil + n$	optimal for $n = 2$; shortest known for $n \geq 6$.
single toggling	$5n - 2$	$5n - 2$	optimal for $n \geq 2$.
general coupling	16, for $n = 2$; $2n^2 + 3n$, for $n > 2$	$2n^2 + 4n$	optimal for $n = 2$; conjectured optimal for $n \geq 3$.
4-limited coupling	$n\lfloor \log_2(n-1)\rfloor$ $+7n$	$4n\lceil \log_2 n\rceil + 17n$	shortest known for $n \geq 15$.
3-limited coupling	$9n - 2$	$14n - 12$	optimal for $n = 2$; shortest known for $n \geq 7$.
2-limited coupling	$9n - 2$	$12n - 8$	optimal for $n = 2$; conjectured optimal for $n \geq 3$.
single coupling	$9n - 2$	$10n - 4$	optimal for $n = 2$; conjectured optimal for $n \geq 3$.
toggle-free coupling	$9n - 2$	$15n - 14$	optimal for $n = 2$; shortest known for $n \geq 4$.

8. Probabilistic Testing

In probabilistic testing a random or pseudo-random test sequence is applied to the circuit under test. For an *error threshold* ϵ with $0 < \epsilon < 1$ and a fault model $\mathcal{F}$ one considers the minimal length $t_0(\epsilon)$ of a random test sequence such that the probability of a fault in $\mathcal{F}$ remaining undetected is no greater than ϵ.

Various assumptions can be made about the stochastic source generating the test sequence. To keep the mathematical analysis manageable it should at least be a Markov source. So far, actually only memoryless sources have been considered for this purpose. With a memoryless source $\mathcal{S}$ as input source, a Markov chain $\mathcal{C}(\mathcal{F}, \mathcal{S}, B)$ can be constructed algorithmically from the observer or its reduced version. This Markov chain, called the *Markov diagnoser,* describes the probabilistic testing process. Let μ_t be the probability of reaching a B-decided state by step t. With ϵ as above, $t_0(\epsilon)$ is the smallest t such that $\mu_t \geq 1 - \epsilon$.

A mathematical analysis of the Markov chain $\mathcal{C}(\mathcal{F}, \mathcal{S}, B)$ has been carried out for random access memories only. For bounded faults, that is, faults involving only a bounded number of memory cells—independent of the size of the memory—the following result was proved.

Theorem 8.1 ([8], [7]). *For probabilistic testing for a bounded fault in an n-bit random access memory one has $t_0(\epsilon) = O(n/\epsilon)$ (under some mild conditions). However, there are bounded faults, namely 2-limited toggling faults, which require a deterministic test length of $\Omega(n \log n)$.*

This result shows that probabilistic testing leads to a significant reduction of the test length when large memories are to be tested and the fault model contains certain complicated faults. A detailed discussion of the merits of probabilistic testing is provided in [8], [14].

Note that the lower bound of $\Omega(n \log n)$ mentioned in Theorem 8.1 is tight (see Table 7.1). The observer and automata derived from the observer turn out to be valuable tools for obtaining such bounds.

9. Conclusions

As diagnosis and testing of VLSI circuits is very complex, a rigorous framework for fault modelling, test sequence generation, complexity analysis of various testing algorithms, and the evaluation of randomized test methods is indispensable. It has been demonstrated several times that automaton and language theory provides that framework. In spite of considerable progress many interesting and challenging problems have yet to be solved.

References

[1] J. A. Brzozowski and B. F. Cockburn, Detection of Coupling Faults in RAMs, *Journal of Electronic Testing: Theory and Applications* 1 (1990), 151–162.

[2] J. A. Brzozowski and H. Jürgensen, Deterministic Diagnosis of Sequential Machines, *Proceedings, 2nd Conference on Automata, Languages, and Programming*

Systems, Salgótarján, Hungary, F. Gécseg and I. Peák, eds., Department of Mathematics, Karl Marx University of Economics, Budapest **1988-4** (1988), 35–49.

[3] J. A. Brzozowski and H. Jürgensen, Probabilistic Diagnosis of Sequential Machines, *Proceedings, 2nd Conference on Automata, Languages, and Programming Systems*, Salgótarján, Hungary, F. Gécseg and I. Peák, eds., Department of Mathematics, Karl Marx University of Economics, Budapest **1988-4** (1988), 51–66.

[4] J. A. Brzozowski and H. Jürgensen, A Model for Sequential Machine Testing and Diagnosis, *Journal of Electronic Testing: Theory and Applications* 3 (1992), 219–234.

[5] J. A. Brzozowski and H. Jürgensen, Component Automata and RAM Faults, *Words, Languages, and Combinatorics II.* Edited by M. Ito and H. Jürgensen, World Scientific, Singapore, to appear (also Report 324, Department of Computer Science, The University of Western Ontario, 1992).

[6] B. F. Cockburn, Fault Models and Tests for Coupling Faults in Random-Access Memories, *Ph.D. Thesis, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, October 1990 (also Research Report CS-90-41, University of Waterloo).*

[7] B. F. Cockburn and J. A. Brzozowski, Near-Optimal Tests for Classes of Write-Triggered Coupling Faults in RAMs., *Journal of Electronic Testing: Theory and Applications* 3 (1992), 251–264.

[8] R. David, J. A. Brzozowski and H. Jürgensen, Random Test Length for Bounded Faults in RAMs, *Proceedings, ETC'93 (European Test Conference)*, IEEE Computer Society Press, Los Alamitos, 1993, 149–158 (also Report CS-92-30, Department of Computer Science, University of Waterloo, 1992).

[9] A. J. van de Goor, Testing Semiconductor Memories. Theory and Practice, *Wiley, Chichester*, 1991.

[10] A. J. van de Goor and B. Smit, Automatic Verification of March Tests, *Records of the 1993 IEEE International Workshop on Memory Testing*, IEEE Computer Society Press, Los Alamitos, 1993, 131–136.

[11] F. C. Hennie, Fault-Detecting Experiments for Sequential Circuits, *Proceedings, 5th Annual Symposium on Switching Theory and Logical Design, IEEE* (1964), 95–110.

[12] F. C. Hennie, Finite-State Models for Logical Machines, *John Wiley & Sons, New York*, 1968.

[13] H. Jürgensen and P. Wong, How to Prove Fault Coverage—or: Testing for the Most Difficult Fault, *Proceedings, 6th Workshop on New Directions for Testing* (May 20–22, 1992), 285–294, *Montréal.*

[14] A. Kraśniewski and K. Gaj, Is There Any Future for Deterministic Self-Test of Embedded RAMs? *Proceedings, ETC'93 (European Test Conference), IEEE Computer Society Press, Los Alamitos* (1993), 159–168.

[15] J. F. Poage and E. J. McCluskey, Derivation of Optimum Test Sequences for Sequential Machines, *Proceedings, 5th Annual Symposium on Switching Theory and Logical Design, IEEE* (1964), 121–132.

[16] L. Shen and B. Cockburn, An Optimal March Test for Locating Faults in DRAMs, *Records of the 1993 IEEE International Workshop on Memory Testing, IEEE Computer Society Press, Los Alamitos* (1993), 61–66.

[17] S. M. Thatte and J. A. Abraham, Testing of Semiconductor Random Access Memories, *Digest of Papers, 7th International Conference on Fault-Tolerant Computing, Los Angeles* (June 28–30, 1977), 81–87.

[18] P. Wong, A Language Theoretic Approach to Fault Coverage and Fault Testing, *M.Sc. Thesis, The University of Western Ontario, London, Ontario, Canada, 1992*

as above, we define a bir
number, modulo 2, of w_a^i
fault type $M^{ia \Rightarrow j}$ we defir

$$\pi^{ia} = \langle$$

For example, let $n =$

u_1 is empty,
$u_2 = r^2 w_0^3 r^3 w_1^3 r_1 w_1^4$
$u_3 = w_0^1 r^1 w_1^4 r^3 w_1^2 w$
$t' = w_0^2 r^2$.

The corresponding vector

$$e^{10} = (0, 1,$$
$$e^{21} = (1, 0,$$

Suppose now that

$$M =$$

Then we have

$$\pi^{10} = 0,\ \pi^{11} = 1$$

Now let

where the summation is t

Lemma 7.1 ([1], [6]) *non-repetitious test t deci* *is not the null vector.*

Continuing with the
Hence, the sequence t in t
machines.

Lemma 7.1 is the ke
length of test sequences fo

(also Report 314, Department of Computer Science, The University of Western Ontario).

[19] M. YANNAKAKIS and D. LEE, Testing Finite State Machines, *Proceedings, 23rd Annual ACM Symposium on Theory of Computing ACM* (1991), 476–485, *New York.*

[20] J. A. BRZOZOWSKI and H. JÜRGENSEN, Further results on the composition of fault types are reported in: J.A. Brzozowski, H. Jürgensen, An Algebar of Multiple Faults in RAMs. Technical Report CS-95-18, Department of Computer Science, University of Waterloo, May 1995.

J. A. BRZOZOWSKI
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF WATERLOO
WATERLOO, ONTARIO
CANADA N2L 3G1

H. JÜRGENSEN
DEPARTMENT OF COMPUTER SCIENCE
THE UNIVERSITY OF WESTERN ONTARIO
LONDON, ONTARIO
CANADA N6A 5B7