A Realistic Fault Model and Test Algorithms for Static Random Access Memories

ROB DEKKER, FRANS BEENKER, MEMBER, IEEE, AND LOEK THIJSSEN

***Abstract*—Testing static random access memories (SRAM's) for all possible failures is not feasible. We have to restrict the class of faults to be considered. This restricted class is called a fault model. A fault model for SRAM's is presented based on physical spot defects, which are modeled as local disturbances in the *layout* of an SRAM. Two linear test algorithms are proposed, that cover 100% of the faults under the fault model. A general solution is given for testing word oriented SRAM's. The practical validity of the fault model and the two test algorithms are verified by a large number of actual wafer tests and device failure analyses.**

I. Introduction

MANY RAM test algorithms based on different fault models have been proposed during the past ten years [1]–[9]. Most of these algorithms were written from a purely mathematical point of view and there is hardly any insight about their practical importance.

The main objective of this paper is to show the feasibility of fault model and test algorithm development based on actual device defects. The defects are modeled as local disturbances in the layout of a static random access memory (SRAM) memory array and translated to defects in the corresponding transistor diagram. The electrical behavior of each defect is analyzed and classified, resulting in a fault model at SRAM cell level. The defect modeling at layout level and extraction to SRAM cell level is done using the inductive fault analysis (IFA) technique as proposed at Carnegie Mellon University [10], [11]. Only single defects per memory cell are assumed.

Two categories of defects can be distinguished at layout level:

1) *global defects*, like too thick gate oxide or too thin poly silicon caused by process etching errors or oven temperature variations;
2) *spot defects*, like dust particles on the chip or the masks, scratches, and gate oxide pinholes.

In this paper only spot defects will be considered. These defects may result in complete or nearly breaks and shorts in the circuit. This paper will only deal with complete shorts and opens. Nearly breaks and shorts generally only cause parametric faults [12]. One of the other objectives of our work was to develop an excellent test algorithm for SRAM self-test applications [13]. Therefore, much effort has been spent to keep the test algorithm regular, symmetric, linear, and with a simple address order. Minimization of the test length was considered a second priority. The test length might be optimized for other applications while maintaining the same fault coverage. RAM's are commonly available in size of N words with m bits per word. In most papers presenting test algorithms only the case of $m = 1$ (a bit oriented SRAM) is considered. We present a general optimal solution to solve the test problems of both bit oriented and word oriented SRAM's.

Manuscript received October 25, 1988; revised July 10, 1989. This paper was recommended by Guest Editor J. Rajski.

R. Dekker and F. Beenker are with Philips Research Laboratories, 5600 JA Eindohoven, The Netherlands.

L. Thijssen is with the Department of Electrical Engineering, Delft University of Technology, Delft, The Netherlands.

IEEE Log Number 9034766.

II. The SRAM Fault Model

For the development of a fault model, an SRAM is divided into three blocks:

1) the memory array;
2) the address decoder;
3) the R/W logic.

These blocks differ in structure, hence, they are analyzed separately. Defects in the address decoder and the R/W logic are *mapped onto* functionally equivalent faults in the memory array. This method, as proposed by Nair, Thatte, and Abraham [4], has the advantage that all faults can be considered to be in the memory array.

In the next sections the following definitions are used.

Definition 1: A memory cell is said to be *stuck-at* if the logic value of the cell cannot be changed by any action on the cell or by influences from other cells.

Definition 2: A memory cell is said to be *stuck-open* if it is not possible to *access* the cell by any action on the cell.

Definition 3: A memory cell with a *transition* fault fails to undergo at least one of the transitions $0 \rightarrow 1$ or $1 \rightarrow 0$.

Definition 4: A memory cell, say cell i, is said to be *state coupled* to another memory cell, say j, if cell i is fixed at a certain value $x (x \in \{0, 1\})$ only if cell j is in one defined state $y (y \in \{0, 1\})$. State coupling is a nonsymmetric relation.

Definition 5: A memory cell i is said to have a *multiple access fault* to another memory cell j if a WRITE action with value $x (x \in \{0, 1\})$ to cell i also forces a WRITE action to cell j with value x. Multiple access is a nonsymmetric relation.

Definition 6: There is said to be a *data retention fault* in a cell if the cell fails to retain its logical value after some units of time.

2.1. *The Memory Array*

The layout of a Philips 8k8 SRAM was used as a vehicle to perform the fault model study. This memory is made in a double poly CMOS process with passive pull-up resistors. In order to model faults in the memory array, a physical defect model has been adapted. We used the spot defect model based on extra and missing layer material as proposed by the IFA method [10], [11]. The model resulted in five different types of spot defects at layout level: broken wires, shorts between wires, missing contacts, extra contacts, and newly created transistors.

Defect analysis is done in two steps. The first step is the translation of spot defects in the layout to defects in the transistor circuit diagram. Fig. 1 shows two examples of spot defects at the layout extracted to circuit level. A detailed analysis of the Philips 8k8 SRAM resulted in at set of 60 spot defects in the layout that were analyzed in more detail. Many different layout defects showed the same defect at transistor diagram level. The second step is to classify defects at transistor level based on equivalent faulty memory cell behavior. The result is a fault model at SRAM cell level, consisting of six fault classes:

1) a memory cell is *stuck-at 0* or *stuck-at 1*;
2) a memory cell is *stuck-open*;
3) a memory cell suffers from a *transition* fault;
4) a memory cell is *state coupled* to another cell;
5) there is a *multiple access fault* from one memory cell to a memory cell at another address;
6) a memory cell suffers from a *data retention fault* in one of its states. The retain time depends on the leak current to the substrate and the capacitance of the floating node. The retain time for the Philips 8k8 design can be up 100 ms.

The complete extraction of the fault model from the circuit defects is given in [14]. A probability factor for a fault class is required in order to find its importance. The probability of a fault caused by a spot defect depends on the *critical area.* This is the chip area where a spot defect can damage the active part of the layout. It depends on the dimensions of the spot defect and the topology of the layout. For example, if two parallel metal wires are separated by 4 μm, then a complete short between these two can only be caused by a spot defect with a diameter of more than 4 μm. The probability of occurrence of the short is proportional to the length of the two wires. This length is called the *critical path length.* The accumulated critical path lengths for each fault class and the minimal spot defect dimensions have been calculated (c.f. Fig. 2). Stuck-at faults have the highest probability of occurrence, but the other fault classes are not negligible. It can be concluded that stuck-at faults only include about 50% of all faults. This is a clear indication that the stuck-at fault model is insufficient for SRAM's.

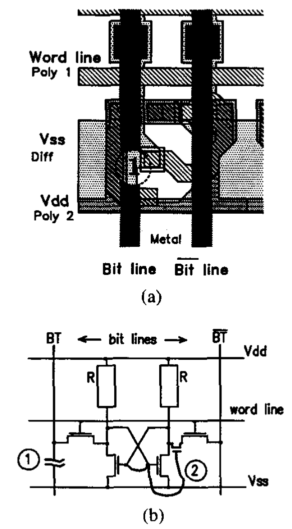


Fig. 1. Layout defects translated to transistor circuit defects according to the IFA method.

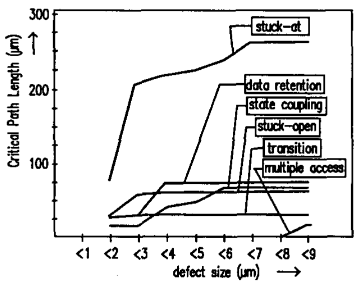


Fig. 2. The critical path length, a measure of the probability of each fault class, expressed against the size of occurring defects.

The critical path lengths, as presented in Fig. 2, were based on the layout of the Philips 8k8 memory cells. Another layout will certainly change the critical path lengths of the various fault classes. However, since the *structure* of SRAM memory cells in various processes is similar, we believe that no *new* fault classes will be introduced in case an other layout or process is used.

2.2. *Address Decoder*

A general fault model for the address decoder is assumed. It was presented by Nair, Thatte, and Abraham [4]:

1) more than one cell is accessed by one address,
2) an address accesses no cells.

Faults in the decoder can be viewed as memory array faults. Item 1 is equivalent to a *multiple access fault* from one cell to one or more cells at other addresses. Item 2 is equivalent to a *stuck-open* cell.

2.3. *R/W Logic*

The R/W logic passes the data information from the I/O pins to the memory array and vice versa. A word oriented SRAM with m bits per word is considered. Faults in the buses, sense amplifiers, and write buffers were considered, resulting in the following fault classes:

1) one or more of the m bits is *stuck-at*;
2) one or more of the m bits is *stuck-open*;
3) a pair of bits is *state coupled.*

All faults in the R/W logic can be regarded as faults in the memory array [4]. Item 1 is equivalent to a set of *stuck-at* cells. Item 2 is equivalent to a number of *stuck-open* cells. Item 3 is equivalent to a *state coupling* fault between two cells at the same address.

III. Design Dependent Conditions

Translation of all faults to the memory array is not sufficient. Detection of all faults is only assured if the R/W logic is *transparent* for all faults in the memory array. This means the R/W logic must pass faults in the memory array to the output pins.

If the sense amplifier is combinational or if the input of the sense amplifier is only one bit line, the sense amplifier will pass a proper defined logical value to the output pin. In this case, a stuck-open fault will be detected as if it were a stuck-at fault.

However, some designs of sense amplifiers include a *data latch* in the read path. This data latch is used to broaden the read window of the RAM during normal operation. It makes the R/W logic sequential. This latch has serious consequences for the fault coverage of cell stuck-open faults since the last read data of the memory will be stored in the data latch. In case a stuck-open cell is read, the contents of the data latch will be passed to the output pin. If this is the expected data to be read from the memory, the stuck-open fault will not be detected.

IV. The SRAM Test Algorithm

Definition 7: A *march element* is a finite sequence of read and/or write operations applied to every cell in the memory array, either in increasing or decreasing address order.

In the proceeding sections, test algorithms will be presented for both bit oriented and word oriented SRAM's with combinational or sequential R/W logic.

4.1. Bit Oriented SRAM with Combinational R/W Logic

A length $9N$ test algorithm is presented, where N is the number of addresses. In order to detect the data retention faults, a data retention test is added. It is given in Fig. 3. A Rd(0) instruction means reading from the memory array and expect the logical 0 from the addressed cell. A Wr(0) instruction means write a logical 0 to the addressed cell. The address is indicated in the first column of the figure.

The proposed wait time in the data retention test depends on the node capacitance and the leakage current in a memory cell. In the Philips 8k8 a wait time of 100 ms was estimated. Other cell designs or processes may result in other wait times.

It can be proven that the $9N$ test algorithm detects all faults of the fault model. Details of the proof can be found in [17].

4.2. Extension for Sequential R/W Logic

If the R/W logic includes a data latch, detection of stuck-open faults is not guaranteed by the $9N$ test algorithm as was explained in Section III. We have solved this problem by adding one extra read action to each march element such that the expected read value is alternating high and low. With this it is ensured that the expected read value is always different from the latest read value. Thus each stuck-open fault will be detected. The resulting $13N$ test algorithm is given in Fig. 4. The data retention test is added. We do not guarantee that this test algorithm is the smallest one possible for 100% fault coverage over the fault model, but it is regular and symmetric, and thus suitable for BIST applications.

4.3. Extension for a Word Oriented SRAM

A read or write action for a word oriented SRAM involves reading or writing an entire word of data, called *data background.* The instructions in the test algorithms (c.f. Figs. 3 and 4) must thus be redefined:

$$\begin{aligned}\mathrm{Wr}(0) &:= \mathrm{Wr}(\langle\text{data background}\rangle)\\ \mathrm{Rd}(0) &:= \mathrm{Rd}(\langle\text{data background}\rangle)\\ \mathrm{Wr}(1) &:= \mathrm{Wr}(\langle\text{inverted data background}\rangle)\\ \mathrm{Rd}(1) &:= \mathrm{Rd}(\langle\text{inverted data background}\rangle).\end{aligned}$$

Word oriented SRAM's introduce the problem of state coupling faults between two cells at one address (or two bits in the R/W logic). To detect those faults all states (00, 01, 10, and 11) of two arbitrary cells at the same address must be checked. This is only possible if several data backgrounds are used in the test algorithm.[1] If m bits per word are used, a minimum of K data backgrounds will be needed where:

$$K = \lceil \log_2 m \rceil + 1$$

$$\lceil x \rceil := \min \{ n \in Z | n \geq x \}.$$

The proof of this statement can be found in [14]. If m is a power of two, the formula simplifies to

$$K = \log_2 m + 1.$$

As an example, we will give the data backgrounds for $m = 8$ (a byte oriented SRAM). According to the formula, K will be 4:

Data background 1	00000000
Data background 2	01010101
Data background 3	00110011
Data background 4	00001111.

[1]Detection of all coupling faults between two cells at the same address is not ensured by the $9N$, $13N$ or any other test algorithm with only one data background. The proof of detection of state coupling faults, only holds for cells at *different* addresses.

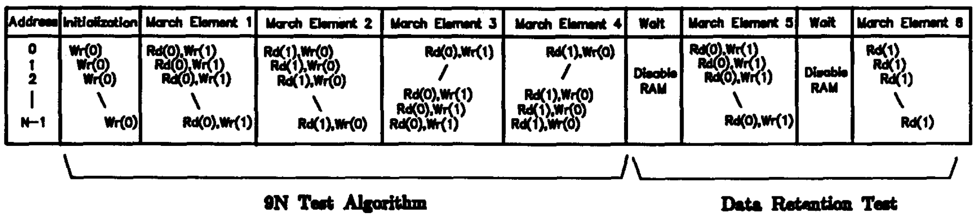


Fig. 3. The 9*N* test algorithm for SRAM's with combinational R/W logic. A data retention test is added.

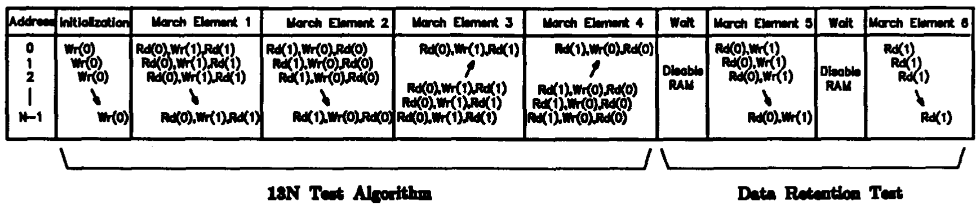


Fig. 4. The resulting 13*N* test algorithm for SRAM's with sequential R/W logic. A data retention test is added.

Detection of all coupling faults is guaranteed, since in the test algorithm the inverted data backgrounds are also generated.

The number of data backgrounds can be minimized if the topology of a particular design is known.

V. Practical Validation

The theoretical results obtained in the previous section have been validated by testing and analyzing a large number of devices of a Philips' 8k8 SRAM. Defective memories have been evaluated and analyzed using microscopes and SEM techniques. In this way the approach to come to a fault model, via a layout study, was validated. Most analyzed devices indeed suffered from the defects as covered by our fault model. Only 10% of the faulty devices behaved in an unexplainable way. A detailed description of the practical validation analysis is given in [14].

5.1. The Objectives

The objectives of this validation phase were twofold.

1) Validation of the fault model. Do the defects, as encountered in our SRAM's, indeed behave as described in the fault model?

2) Validation of the test algorithms. How does our test behave in comparison with test algorithms as proposed in literature?

5.2. The Test Data Analysis

1192 devices of the Philips 8k8 SRAM were taken out of 9 wafers from 3 different batches. These devices were tested with a set of 23 test algorithms on a Teradyne J389 memory tester. This set of algorithms consists of the algorithms derived in Section II and a number of algorithms as proposed in literature. The choice of the latter algorithms were guided by a previously performed extensive algorithm-comparison study [15]. Finally, various implementations of the algorithms were used based on several addressing orders and data backgrounds.

Executing a test algorithm on one device results in a number of detected faulty cells. This number is called the *bit error number.* It is dangerous, however, to rely completely on the bit error number as a measure of test algorithm effectiveness. One defect in a decoder may result in a large bit error number. One defect in the memory array may result in a small bit error number. A second problem with the bit error number is the interpretation of the defects. The test algorithms used do have an overlap with respect to detection capabilities of the faults in our fault model. Correlations will, therefore, exist between the different bit error numbers of the various test algorithms.

The bit error number can vary strongly from algorithm to algorithm and from device to device. We have translated defects in the decoders and the R/W logic into equivalent faults in the memory array. Hence, it is possible that, although for various devices the bit error number is completely different, the actual fault class is the same.

In order to find these correlations the "behavior" of the test algorithms over one device and the "behavior" of the test set over the total number of devices has been computed. The behavior of the test algorithms over one device is described by using the notion of *signature.* The behavior of the test sets over the total number of devices is described by the notion of a *cluster.* The signatures were also used to measure the effectiveness of the test algorithms.

For each device, each test algorithm is assigned a signature bit with values 0, 1 or 2. This bit is a function of the bit error number of the corresponding test algorithm.

cluster	*#devices*	*fault class*
0	714	cell stuck-at & total chip failure
1	169	cell stuck-open faults
2	18	multiple access faults
3	9	state coupling faults
4	8	?
5	5	?
6	26	cell data retention faults
—	—	?
14	2	?

(a) (b)

Fig. 5. (a) Resulting chapters with (b) observed fault class.

The value of the signature bit is determined according to the following rules:

$m - r/5$

$$\text{bit error number} < m - r/5 \rightarrow \text{sign. bit} = 0$$

$$< \text{bit error number} < m + r/5 \rightarrow \text{sign. bit} = 1$$

$$\text{bit error number} > m + r/5 \rightarrow \text{sign. bit} = 2.$$

where:

m mean value of the bit error numbers for one device,

r standard deviation of the bit error numbers for one device.

This process resulted into a 23 bit signature per device which may be considered as a fault-class representative.

Some of the most frequently occurring signatures differed at one or two signature bits. The differences in these signatures were caused by random defects and multiple faults on a device. To obtain the main fault classes the signatures were clustered using a statistical clustering algorithm, similar to the *isodata* algorithm developed by Tou and Conzalez [16]. We expected that each cluster could be characterized by a single fault class.

5.3. Validation of the Fault Model

Our analysis resulted into 15 clusters (c.f. Fig. 5(a)). Several devices from each cluster were analyzed in more detail using a light microscope and a SEM. Defects in devices from the same cluster indeed all suffered from the same fault class. The result of this analysis is shown in Fig. 5(b). Notice, that most of the theoretically derived fault classes are indeed important and were encountered in practice. Only transition faults were hardly ever found and multiple access faults occurred regularly because of faults in the decoder. In [17] examples of spot defects in devices from several clusters are shown.

About 10% of the analyzed defects showed a not yet understood behavior. This kind of fault remains a point of further research and improvement of the IFA technique and fault model.

5.4. Validation of the Test Algorithm

The signatures used for the fault model validation analysis were also used for the test algorithm validation analysis. Each algorithm was assigned a *score number.* This number is the mean value taken over the signature bits from all devices of one test algorithm. Therefore, the maximum score that could be reached for a test algorithm was 2. The higher the score number of a test algorithm, the higher its fault coverage of real faults compared to other test algorithms. The results of the test algorithm comparison study is shown in Fig. 6.

The test algorithms as developed in Section IV are called IFA_9*N* and IFA_13*N*.

lit. ref.	*address order*	*algorithm name*	*# dbgr*	*data ret. test*	*Score (%) number*
3	X	MATSPLS	1	NO	7
3	Y	MATSPLS	1	YES	18
3	JUMP	MATSPLS	4	NO	53
9	JUMP	MARIN_C	1	YES	75
9	X	MARIN_C	4	NO	61
9	Y	MARIN_C	4	YES	89
9	X	MARIN_B	1	YES	82
9	X+Y	MARIN_B	4	NO	83
1	X	MRCH_01	1	YES	34
1	Y	MRCH_01	4	NO	78
5	Y	SUCK_RD	4	NO	78
5	X+Y	SUCK_RD	1	YES	88
4	X	NAIR.AL	4	YES	84
4	X+Y	NAIR.AL	1	NO	82
2	Y	PAPACHR	1	NO	74
2	X+Y	PAPACHR	4	YES	96
	Y	IFA_9N	4	YES	91
	Y	IFA_13N	4	YES	92
	Y	IFA_13N	4	NO	80
7		PATSENS		NO	39

Fig. 6. The detection properties (score) of all test algorithms over 480 defective devices. The applied algorithms are given plus literature reference, the address order used, the number of data background (dbgr), and an indication whether or not a data retention test was added.

VI. Conclusions

Defect analysis using the IFA technique results into a physical failure-based fault model. The proposed test algorithms show excellent features in both test time and fault coverage. The fault model and test algorithms have been validated in practice. The test algorithms cover 100% of the faults under the fault model and are independent of row, column, and cell arrangement in the memory array. The choice of the test algorithm is slightly design dependent. A length 9*N* test algorithm is given for SRAM's with combinatorial R/W logic. A length 13*N* test algorithm is given for SRAM's with sequential R/W logic. The given test algorithms are both suitable for bit and word oriented SRAM's. We have taken advantage of the regularity and symmetry of the test algorithm by implementing them in a self-test machine [13].

It is important to notice that 60% of the defects behave as stuck-at faults (memory stuck-at faults and total chip failures). The yield of the sample was low, so many total chip failures occurred. Hence, a sample with a higher yield (less total chip failures) will show an even lower percentage of stuck-at faults. In the theoretical approach, single defects were assumed and 50% stuck-at was estimated.

Ninety percent of the analyzed defects could be explained by using the IFA technique and are covered by the fault model. Almost all fault classes as indicated in the fault model indeed occurred regularly in actual devices. The 10% of not yet understood defects form the input of current research activities to achieve improved defect modeling and fault modeling techniques.

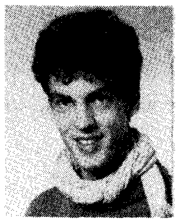
Acknowledgment

The authors would like to thank Dr. William Rey of the Philips Research Laboratories. He largely contributed to the data analysis of the Philips 8k8 SRAM test data.

References

[1] M. Abadir, H. Reghbati, "Functional testing of semiconductor random access memories," *Ass. Comput. Math Comput. Surveys*, vol. 15, no. 3, pp. 174-198, Sept. 1983.
[2] C. Papachristou and N. Sahgal, "An improved method for detecting functional faults in semiconductor random access memories," *IEEE Trans. Comput.*, vol. C-34, pp. 110-116, Feb. 1985.
[3] R. Nair, "Comments on "An optimal algorithm for testing stuck-at faults in random access memories," *IEEE Trans. Comput.*, vol. C-28, pp. 258-261, Mar. 1979.
[4] R. Nair, S. Thatte, and J. Abraham, "Efficient algorithms for testing semiconductor random-access memories," *IEEE Trans. Comput.*, vol. C-27, pp. 572-576, June 1978.
[5] D. Suk and S. Reddy, "A march test for functional faults in semiconductor random access memories," *IEEE Trans. Comput.*, vol. C-30, pp. 982-985, Dec. 1981.
[6] S. Thatte and J. Abraham, "Testing of semiconductor random access memories," in *Proc. 7th Annu. Int. Conf. on Fault-Tolerant Computing*, June 1977, pp. 81-87.
[7] J. Hayes, "Detection of pattern sensitive faults in random access memories," *IEEE Trans. Comput.*, vol. C-24, pp. 150-157, Feb. 1975.
[8] V. Scrini, "API Tests for RAM Chips," *IEEE Computer*, vol. 10, pp. 32-35, Jan. 1977.
[9] M. Marinescu, "Single and efficient algorithms for functional RAM testing," in *Proc. IEEE Int. Test Conf.*, 1982, pp. 236-239.
[10] J. Shen, W. Maly, F. Ferguson, "Inductive fault analysis of CMOS integrated circuits," *IEEE Design Test Comput.*, pp. 13-26, Dec. 1985.
[11] W. Maly, "Modeling of lithography related yield losses for CAD of VLSI circuits," *IEEE Trans. Computer-Aided Design*, vol. CAD-4, pp. 166-177, July 1985.
[12] M. Syrzycki, "Modelling of spot defects in MOS transistors," in *Proc. IEEE Int. Test Conf. 1987*, paper 6.1, 1987, pp. 148-157.
[13] R. Dekker, F. Beenker, and L. Thijssen, "A realistic self-test machine for word oriented static random access memories," in *Proc. IEEE Int. Test Conf. 1988*, paper 20.2, 1988, pp. 353-361.
[14] R. Dekker, "Fault modeling and self-test of static random access memories," TUD Rep. 1-68340-28(1987)25, Dep. Elect. Eng., Delft Univ. of Technology, The Netherlands, 1987.
[15] P. Veenstra, F. Beenker and J. Koomen, "Testing of random access memories, theory and practice," *Inst. Elect. Eng. Proc.*, vol. 135, pt. G, no. 1, pp. 24-28, Feb. 1988.
[16] J. Tou and R. Gonzalez, *Pattern Recognition Principles.* Reading, MA: Addison-Wesley, 1974, pp. 90-104.
[17] R. Dekker, F. Beenker, and L. Thijssen, "Fault modelling and algorithm development for static random access memories," in *Proc. IEEE Int. Test Conf. 1988*, paper 20.1, 1988, pp. 343-352.

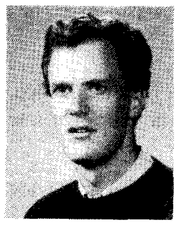
*



Rob Dekker received the M.S. degree in electrical engineering from the Delft University of Technology, The Netherlands, in 1987.

Currently, he is a member of the research staff of Philips Research Laboratories, where his present work involves DFT for a DSP-oriented silicon compiler. His research interests include design for testability, circuit design, silicon compilers, CAD, CAT, fault modeling, and test pattern generation.

*

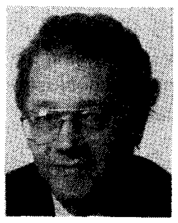


Frans Beenker (M'88) received the M.Sc. degree in mathematics and computing science from the Eindhoven University of Technology, The Netherlands, in 1982.

Currently, he is the test project leader of a DSP-oriented silicon compilation project at Philips Research Laboratories. He is also a member of the CAD for VLSI Systems group at Philips. He is the coauthor or author of many papers in the field of testing.

Mr. Beenker was the Chairman of the 1988 European Design for Testability Workshop.

*



Loek Thijssen received the M.S. degree in electrical engineering from the Delft University of Technology, The Netherlands, in 1968.

Since 1970, he has been a research scientist at the Department of Electrical Engineering, Delft University. His research interests include the field of switching theory and logic design. He is the coauthor of several text books on these areas of research.