Random access memory testing : theory and practice : the gains of fault modelling

THE
Research Report

Eindhoven
University of Technology
Netherlands

Department of Electrical Engineering

Random Access Memory Testing:

Theory and practice

The Gains of Fault Modelling

by
P.K. Veenstra

EUT Report 86-E-161
ISBN 90-6144-161-7
ISSN 0167-9708
October 1986

Eindhoven University of Technology Research Reports

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering

Eindhoven The Netherlands

RANDOM ACCESS MEMORY TESTING: THEORY AND PRACTICE

The Gains of Fault Modelling

by

P.K. Veenstra

EUT Report 86-E-161
ISBN 90-6144-161-7
ISSN 0167-9708
Coden: TEUEDE

Eindhoven
October 1986

CIP-GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Veenstra, P.K.

Random access memory testing: theory and practice. The gains of fault modelling / by P.K. Veenstra. - Eindhoven: University of Technology. - Fig., tab. - (Eindhoven University of Technology research reports / Department of Electrical Engineering, ISSN 0167-9708; 86-E-161)
Met lit. opg., reg.
ISBN 90-6144-161-7
SISO 664.2 UDC 621.382.001.42 NUGI 853
Trefw.: elektronische schakelingen; tests.

ABSTRACT

Functional testing of Random Access Memories gives more and more problems. At the one hand, the dimensions are growing rapidly and the denser devices result in multiple failure modes. At the other hand, the accessibility of embedded memory elements on modern VLSI circuits decreases. The goal of the research was the evaluation and the mutual comparison of the existing methods and ideas in this field, eventually followed by setting up methodologies to improve the testability of embedded RAM's. After the literature has been inventoried, the decision was made to examine the large diversity of test patterns by means of large-scale measurements. Such a session has been executed two times and the results affirm the previous assumptions, namely:

1. the use of a well-defined fault model translates itself into a better fault coverage and more efficient test patterns;
2. time-dependent failures require specific measures.

SAMENVATTING

Het functioneel testen van Random Access Memories leidt in toenemende mate tot problemen. Enerzijds nemen de dimensies steeds grotere vormen aan en veroorzaakt de schaalverkleining een veelvoud van foutenbronnen. Anderzijds ontstaan moeilijkheden met betrekking tot de toegankelijkheid van geïntegreerde geheugenmodulen in VLSI-schakelingen. Doel van de opdracht betrof het evalueren en onderling vergelijken van de reeds voorhanden zijnde methoden en ideeën op dit gebied, alsmede het opstellen van methodieken om de geïntegreerde geheugenelementen beter testbaar te maken, zover als de tijd dit toelaat. Na een inventarisatie van de literatuur werd besloten de grote verscheidenheid aan testpatronen middels grootschalige metingen te onderzoeken. Tot tweemaal toe is een dergelijke meetsessie uitgevoerd en de resultaten bevestigen eerdere veronderstellingen, namelijk:

1. het hanteren van een goed gefundeerd foutmodel vertaalt zich in een betere foutdetectie en efficiëntere testpatronen;
2. de tijdsafhankelijke fouten vereisen specifiekere maatregelen.

Veenstra, P.K.
RANDOM ACCESS MEMORY TESTING: Theory and practice. The gains of fault modelling.
Department of Electrical Engineering, Eindhoven University of Technology, The Netherlands, 1986.
EUT Report 86-E-161

PREFACE

The research was conducted within the scope of the training for the M.Sc. degree in electrical engineering at the Eindhoven University of Technology. The Philips Research Laboratories in Eindhoven offered me the opportunity and had the disposal of the necessary facilities to do the investigations.

The work was carried out within the group "CAD for VLSI systems" in close co-operation with the group "test and analysis".

The appointed task was to draw up an inventory of the existing Random Access Memory testing methods and to compare their properties with respect to fault coverage and complexity.

The time was spent as follows:

from mid September	1985	to mid November	1985:	study of literature
from mid November	1985	to mid February	1986:	setting up test program 1
from mid February	1986	to mid March	1986:	analysis of the results
from mid March	1986	to mid April	1986:	setting up test program 2
from mid April	1986	to end April	1986:	analysis of the results
from beginning May	1986	to end May	1986:	first edition of the thesis
from beginning June	1986	to mid July	1986:	speeches, final thesis, paper.

The work was supervised by:

Prof. ir. A. Heetman and ir. M.J.M. van Weert,
Group of Digital Systems,
Department of Electrical Engineering,
Eindhoven University of Technology, The Netherlands

and

ir. F.P.M. Beenker and Dr. J.J.M. Koomen,
Philips Research Laboratories,
Eindhoven, The Netherlands

TABLE OF CONTENTS

1 COMPONENT TESTING: A SURVEY

1.1 Introduction

As systems grow in complexity the need for systematic test strategies increases too. A failure on a higher system level will on the average result in an increase of factor ten of the costs of repair. Therefore manufacturers of complicated systems attach very great value to highly reliable components.

Up to twenty years ago the producers of semiconducter elements were only concerned with the testing problem of relative simple components like resistors, transistors and diodes. More complicated problems did not occur until a higher system level, the board level, i.e. when the several elements already form part of a functional circuit.

Ten years later the more sophisticated logic systems were still built up out of all sorts of seperate parts as combinatorial logic, shift registers, clock generators, etc. Each component could easily be tested exhaustively because of its simplicity. So, the testing problem of the earlier boards was moved downwards to the component level.

Nowadays the extreme integration leads to peculiar complex circuits. Several hundreds of thousands of elements can be housed on a single IC and even different kinds of functional blocks form the basis of the new generation processor chips. Nevertheless, still a few pins are the only communication means with the external world. This gives rise to increasing difficulties in the testing field with which especially designers and test engineers are faced.

Due to the rapid grow in complexity of modern ICs, designers and production engineers have a growing interest in testing. Methods as Scan Path are introduced. Moreover, the increasing costs of test equipment and longer test times during the production stage, makes that the remaining indifference of designers and production engineers toward testing is rapidly changing. Hence, test considerations become more and more a major theme of design, rather than a simple afterthought.

1.2 Logical Fault Models

A combinational circuit doesn't work correctly if, as result of one or more logical faults, the desired function $f[x(1),x(2),...,x(n)]$ is changed into $g[x(1),x(2),...,x(n)]$. Hence, a set of input values is needed to detect these possible irregularities. Testing for all possible failure mechanisms is not feasible. So, a choice has to be made for

faults most likely to occur. This set of faults is called a fault model and a set of test vectors is needed with respect to the fault model. Such a fault model must also specify the effects of each input vector on the circuit outputs, offering the possibility to verify the responses, i.e. to determine whether a particular failure mode is present or not.

Over the years the single stuck-at model [1] turned out to be very useful with respect to small combinational circuits, see figure 1, despite of its fundamental limitations. For instance, most bridging fauts are not covered, whereas the coverage of multiple faults is unknown. Nevertheless, the Boolean difference, critical path and D-algoritm techniques [2] rest on this assumption. The simplicity and straightforward approach of the single stuck-at model form the reasons as well as the lack of a better alternative.

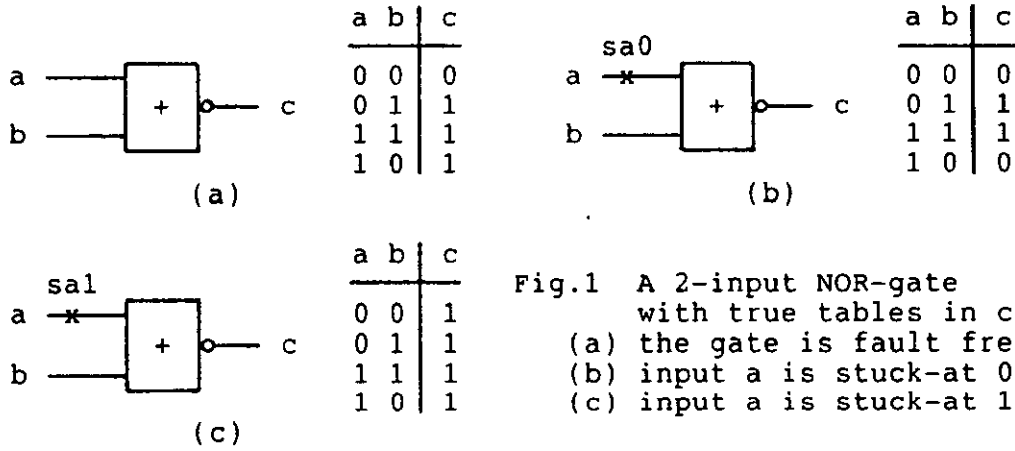


Fig.1 A 2-input NOR-gate with true tables in case (a) the gate is fault free, (b) input a is stuck-at 0 and (c) input a is stuck-at 1.

Large testing problems however, occured in case of complexer combinational and sequential circuits.

1.3 Testing Problems

The combinational circuits give difficulties as a result of the determination of suitable test patterns and the presence of redundancy in the circuits.
Each doubling of the number of logical gates produces an exponentional increase of the test pattern generation complexity. This result is caused by the fact that each possible stuck-at fault has to be propagated in forward and backward direction to get an unambiguous set of input test vectors for which the considered fault will be propagated to the outputs.

On the other hand, testing a PLA with the single stuck-at model gives a very low fault coverage [3]. In general PLAs

possess a high degree of redundancy, so the bad results follow in consequence of the unsuitable fault model. Examination of the physical failures and the internal structure of PLAs has resulted in the introduction of the crosspoint fault model [4]. Until now this model has been adapted as starting point for many reliable PLA test programs.

But for all the sequential circuits gave enormous problems. The nature of this circuitry makes that the logical behaviour had to be tested for each possible state. Apart from the fact that total state controllability is desired, an infeasable number of input patterns will be needed.

Another phenomenon appeared in case of asynchronous designs whereby certain unwanted transitions due to races and hazards couldn't be examined properly.

1.4 Design for Testability

To avoid the testing problems of sequencial circuits, several ideas were proposed varying from the addition of extra pins for increasing controllability and observability to suggestions which result in severe restrictions of the designer's freedom.

The usual way is a design for testability method [5,6] by which the sequential elements function synchronously and have the ability to form long shift registers in the test mode like IBM's LSSD (Level-Sensitive Scan Design) or the Scan Path method used within Philips, see figure 2.
This enables the possibility to partition the sequential circuit into small combinational blocks such that each part can be controlled and observed. In this way the generation and administration of the test patterns becomes quite manageable as well as the examination of the results.

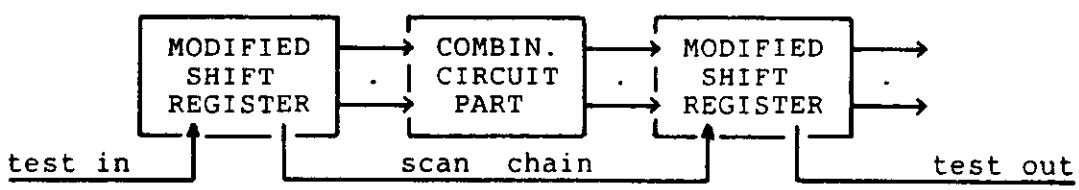


Fig.2 The shift registers are connected in the test mode, enabling the verification of embedded circuit parts.

Still increasing circuit complexities have led to a number of self-test proposals based on the scan design concept. With slight modifications the shift registers can be changed into pseudo random pattern generators and signature

registers [7] to generate random test vectors and to compress the test responses internally. Other self-test techniques use externally located generators and/or registers to simplify testing [6].

In principle, testing and test generation of combinational and scan-testable sequential logic gives no particular problems anymore. Actually, the most important questions concern circuits like PLAs, RAMs, processors, multiplexers, etc., which ask for dedicated test approaches.

1.5 Macro Testing

Housing of combinational logic, PLAs, RAMs, etc. on the same chip seems to be the testing problem of the nearest future, see figure 3. Recently, a test approach called "macro-testing" has been proposed [8]. The approach starts from the idea that these complex systems can be visualized as hierarchical structures with a distinguishable control, memory and I/O part.

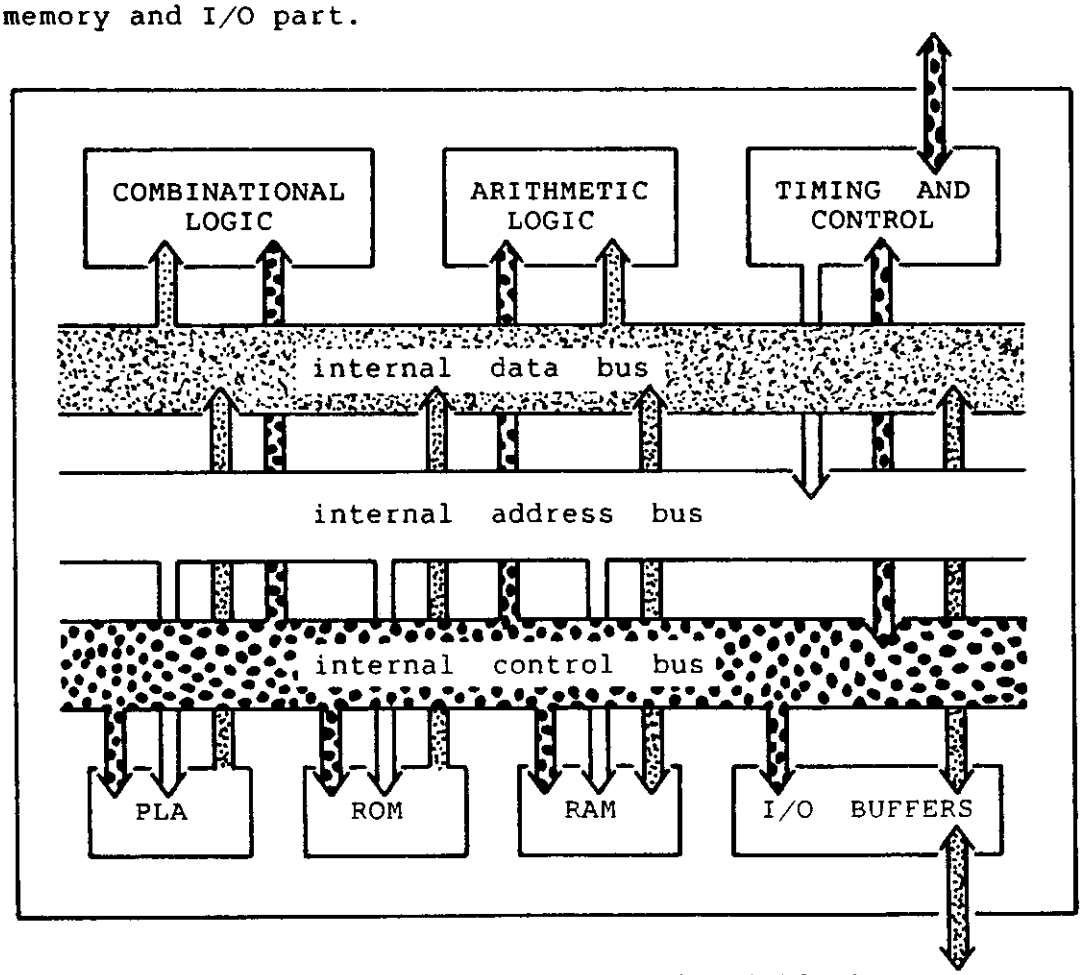


Fig.3 Device containing several functional blocks.

A macro test starts with an extensive examination of each unit, followed by an overall system test. During the test stage a separation has to be forced which enables an individual test of each unit. The test stimuli can be supplied from outside [9] by means of a serial chain or through the addition of self-test circuitry to each functional block [10]. In turn the test ends with a small overall test.
The first step offers the possibility to verify each part for its specific properties in accordance with the most suitable fault model. The approach will result in efficient and by all reliable tests of each building block. In essence the second step is reduced to a check on the interconnections.

1.6 Future Developments

From quite another point of view the IC manufacturers are faced with new demands. The rapid growing interest in custom designed ICs asks for quick turnaround times. Instead of one year or longer an IC has to be designed, fabricated and tested in only a few weeks, see figure 4. The gate-array and standard cell designs are the simple forerunnurs of an important part of the future IC-market [11,12].

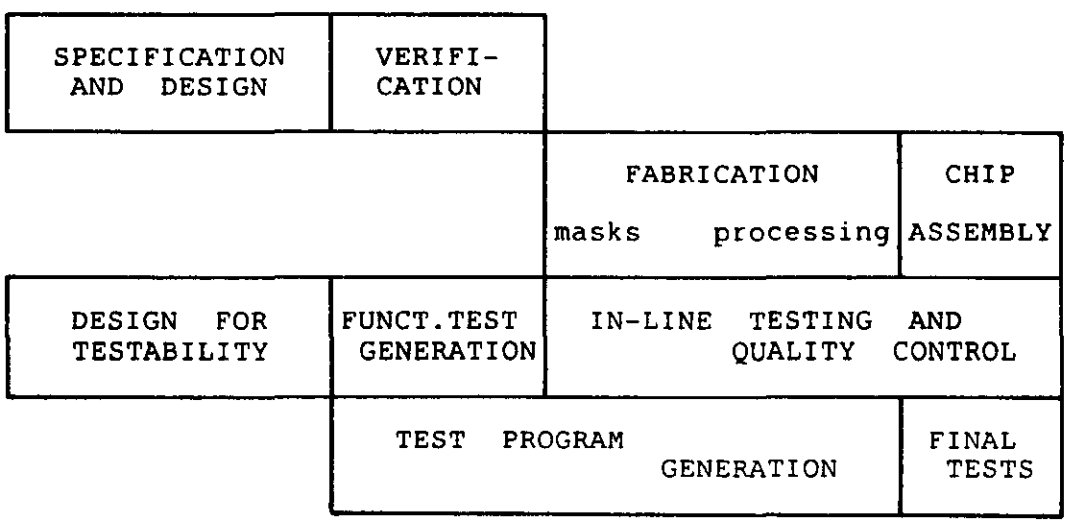


Fig.4 Time-schedule with the integration of designing, fabricating and testing to reach quick turnaround times.

A manufacturer can only handle these requirements through the further development and integration of software and fabrication tools, i.e. CAD/CAM. The final goal is to have the disposal of true silicon compilers which convert a formal description of a logical circuit into a physical

layout and provide with an adequate test program before the fabrication level is entered.

From the standpoint of testing this evolution demands for the gross introduction of strong formal design rules to ensure the testability and correctness of the product at the end. That is the only way by which the test engineeer will be able to write universal test programs and to specify the insertion of test circuitry.
However, setting up a packet of demands is only possible if the drafter has a well understanding of the failure mechanisms. This asks not only for theoretical fault models but also extensive experiments to develop reliable and especially efficient test programs.

1.7 Considerations

The importance of testing is growing rapidly due to an increasing complexity of modern IC designs. More and more the trend can be visualized that during an early stage of the design process attention is payed to the testability of the ultimate product.
In spite of this favourable development the test engineer is confronted with big challenges as result of difficulties like a decreasing accessibility of the circuits to be tested, the introduction of ICs built up out of diverge functional parts, the absence of accurate fault models necessary for the development of efficient test programs, etc. Problems which will become more severe with the next generation custom designed ICs in progress.

These considerations reflect the context in which one part of this report has to be placed, namely the testing problem of the so-called embedded RAMs. At the other hand attention is employed on normal memory devices. As can be expected, this subject has great overlap with that of embedded RAMs.

2 RANDOM ACCESS MEMORIES

2.1 Dynamic versus Static RAMs

2.1.1 Operation of a DRAM Cell

The absence or presence of an electrical charge on small capacitors form the operational basis in dynamic RAMs to define logical levels. A memory cell of the modern high density DRAMs only consists of one small transistor and a very small capacitor [13], see figure 5.a. In each column a single cell can be accessed by activating the corresponding word line. This makes that all capacitors of the cells in that row have become connected with the corresponding bit lines. These bit lines form the connections with the common sense amplifiers, shared between all cells in the same column. In the end, the addressed cell will be selected by means of the column decoder.
In case of a read operation the amplifier determines whether the capacitor was charged or empty, corresponding with the logical 1-value or 0-value, respectively. Simultaneously this investigation is attended by a recovery of the initial cell's content. When a write operation takes place, an arbitrary value can be stored in the memory cell. Leakage currents make it necessary to refresh the electrical charges regularly.

2.1.2 Operation of a SRAM Cell

The static memory cells are constructed as flip-flops, so refresh operations can be omitted. Three design types can be distinguished [13], the passive pull-up type or resistance load SRAM cell, see figure 5.b, as well as two active pull-up types, see figure 5.c and 5.d.
Fast SRAMs are normally based on the six-transistor concepts. The older designs were based on the cell type of figure 5.c, whereas modern devices are based on the full CMOS cell of figure 5.d. These memory cells require more chip area and have a larger energy consumption during switching than their passive counterparts. The two additional transistors are especially added to perform quick changes of the cell content during a transition write operation. In turn, the passive pull-up cells require less silicon area and are used in the high density SRAMs.
Before a memory operation can take place, the bit lines will be precharged. In case of a write operation the driver will set the bit line pair of the addressed memory cell in an unbalanced state by which the accessed cell is forced to flip if it was in the opposite state. Similarly, an addressed cell brings its bit line pair out of balance during a read action, from which the sense amplifier can deduce the polarization of the cell.

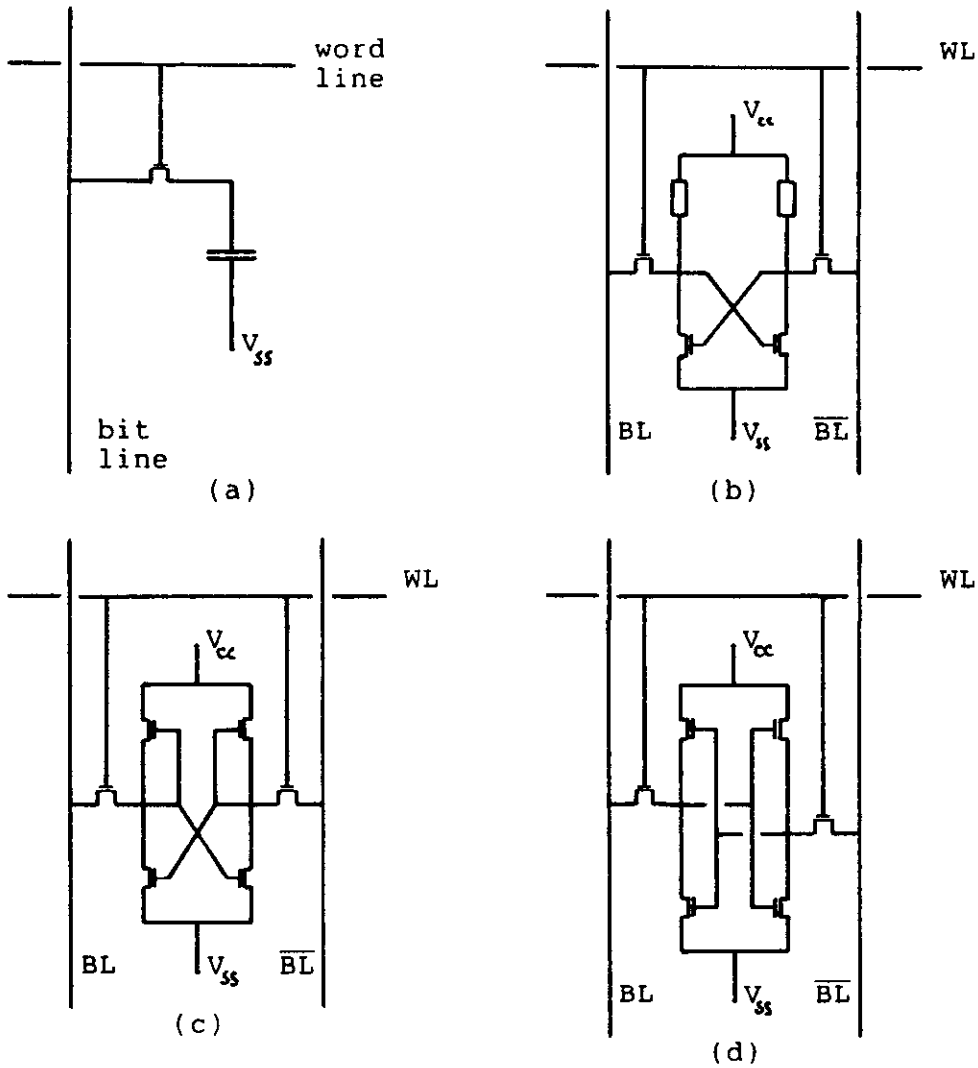


Fig.5 Structures of (a) a DRAM cell, (b) a resistance load SRAM cell, (c) an older active pull-up SRAM cell and (d) the modern full CMOS active pull-up SRAM cell.

2.1.3 Application Areas

The functional difference between a DRAM cell and the passive pull-up SRAM cell makes that the high density DRAMs have about four times the capacity of comparable SRAMs. On the contrary, SRAMs are normally less power consuming and easier to employ.
Dynamic memories are mostly used in mini-processor and micro-processor based systems. The complexity of these systems makes that considerations like power consumption, battery back-up and the additional refresh circuitry are of minor importance, opposite to the desired memory capacity.

The normal SRAM devices can be found mainly in consumer electronics and portable equipment. A static memory doesn't require much additional control circuitry as well as a very small standby power consumption.
The very fast static SRAMs form a special category. Their active pull-up cell configuration is used to construct memories with a relative small number of memory cells but a very good speed performance. They are especially suitable for the cache memories of mainframe computers and other time-critical applications.

2.2 Internal Organization of a RAM

The following basic elements can be distinghuished in a RAM:

1. the memory array part;
2. the address decoders and address registers;
3. the data transfer and read/write logic;
4. circuitry to control the internal timing and energy supply of the chip.

The mutual relations of these functional parts are shown in the block diagram of figure 6. However, the actual chip organization is slightly different due to the design considerations. Two configurations are mainly used. One where the sense amplifiers are embedded in the memory arrays, see figure 7.a [14]. Especially DRAMs have this structure to facilitate quick refresh operations and faster access times. The second organization plan is restricted to SRAM designs, where the sense amplifiers are located between the column decoder and the I/O buffers, see figure 7.b [15].

2.2.1 Memory Array

In modern RAMs the memory part occupies up to 80% of the total chip area. As result of the regular structure very high densities can be obtained requiring extreme process conditions. The array is mostly devided into twofolds of equal sub-arrays which offers the opportunity to share circuitry between several parts. The arrays are built up out of identical cells arranged as a matrix in rows and columns. In the horizontal direction a common word line forms the connection to the row decoder, whereas common bit lines open the possibility to select a certain column. Together the possibility is provided to select a single cell.

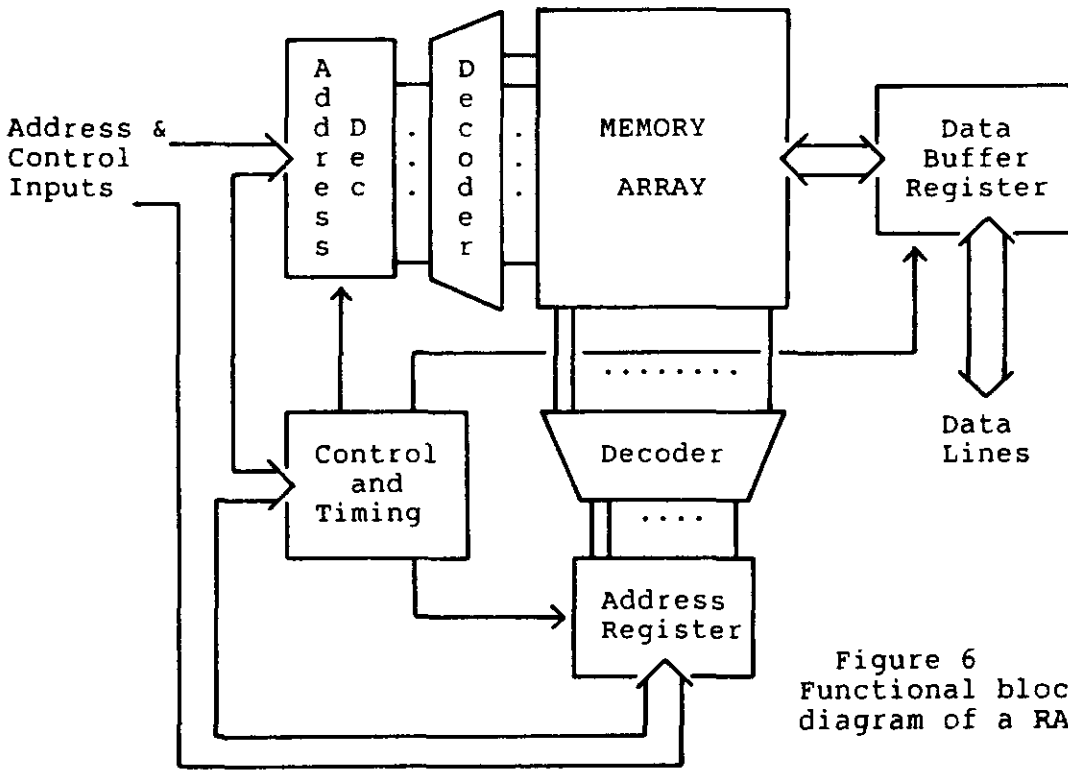


Figure 6
Functional block
diagram of a RAM

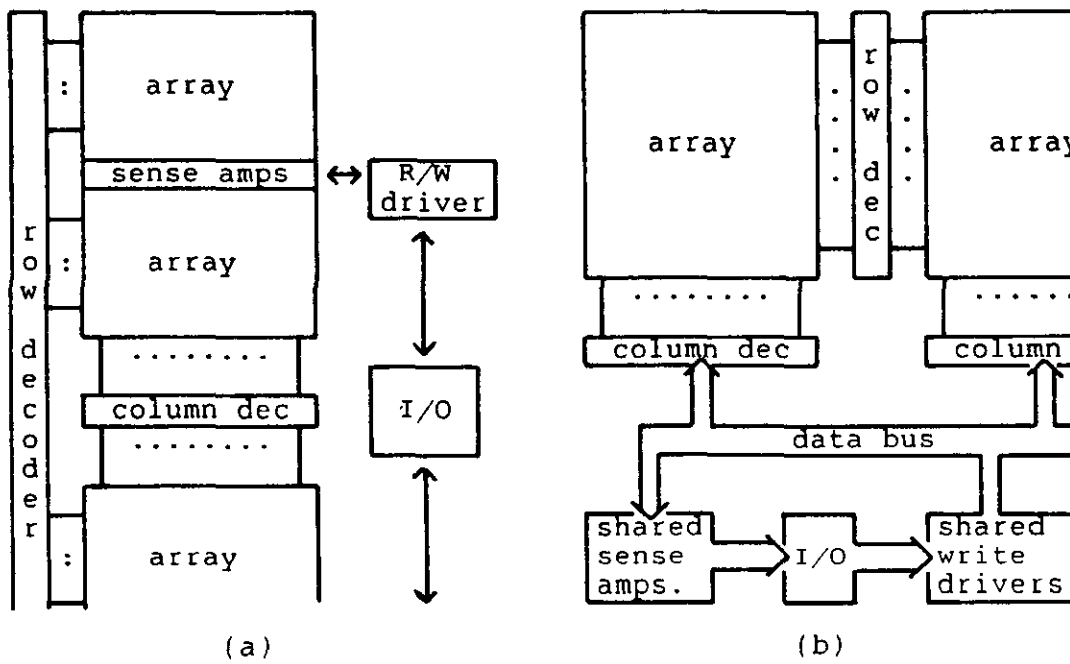


Fig.7 General chip organization of (a) a DRAM with the sense amplifiers embedded in the arrays and (b) a SRAM with the completely shared read and write circuits.

2.2.2 Addressing Mechanism

Each memory position has a corresponding unique address. To activate this position, the address will be stored in the address register. In turn, the row decoder accesses the desired row(s) by activating the corresponding word line(s). Finally, the column decoder establishes the connection between the memory cell(s) and the data logic.
To achieve shorter access times a lot of the DRAMs are equiped with seperate decoder control lines. If one activates the CAS (column address select) line alone, the cells in the same row can be read or write with a faster data rate. This is known as page-mode operation, whereby the row decoder is fixed at one position. Similarly, refresh operations are facilitated with a disabled column decoder, which makes that all the cells in one row can be refreshed by cycling only the RAS (row address select) line. The internal structure of the decoders results in a scrambling of the addresses, i.e. the absence of a direct relationship between logical and topological addresses. This give problems if data patterns are based on the internal structure of the RAM. We will explain this problem in section 2.3.4.

2.2.3 Data Flow

According the DRAM scheme of figure 7.a, the cells in one column have one symmetrical sense amplifier in common. At both sides of this sense amplifier a dummy cell is present, storing a logical 0 value. The dummy cell is approximately half the size of a normal DRAM cell and the proper dimension of this cell forms the most critical part of a DRAM, because of its reference task. Figure 8 represents the organization of the memory cells around the common sense amplifier.

Before an operation can take place, both bit line halves are precharged by activating Q_1 and the Q_2's. In turn they are separated and made floating at precisely the same voltage due to a deactivation of Q_1 followed by the Q_2's.
When a read action is performed, the activated word line connects the cell capacitor to the corresponding bit line half, thus to one side of the sense amplifier, whereas simultaneously the dummy cell will be accessed on the opposite side. If the cell stores a zero, its empty capacitor will drop the potential of the corresponding bit line half below the potential of the opposite bit line half set by the dummy cell. Similarly, a logical one leads to an opposite result. The amplifier is enabled with the Q_3's and sensitizes this voltage difference, which in turn is amplified by discharging the bit line with the lowest level: an action performed by a slow pull down of the precharged node (a). This ensures also that the operation of the sense amplifier is insensible to variations of the charges in the

cells. After that the data (i.e. the voltage difference) is transferred to the output buffer via the I/O lines due to an activation of the Q_4's.
In case of a write operation, the data on the I/O lines can simply be stored in the accessed RAM cell by means of the corresponding bit line half.

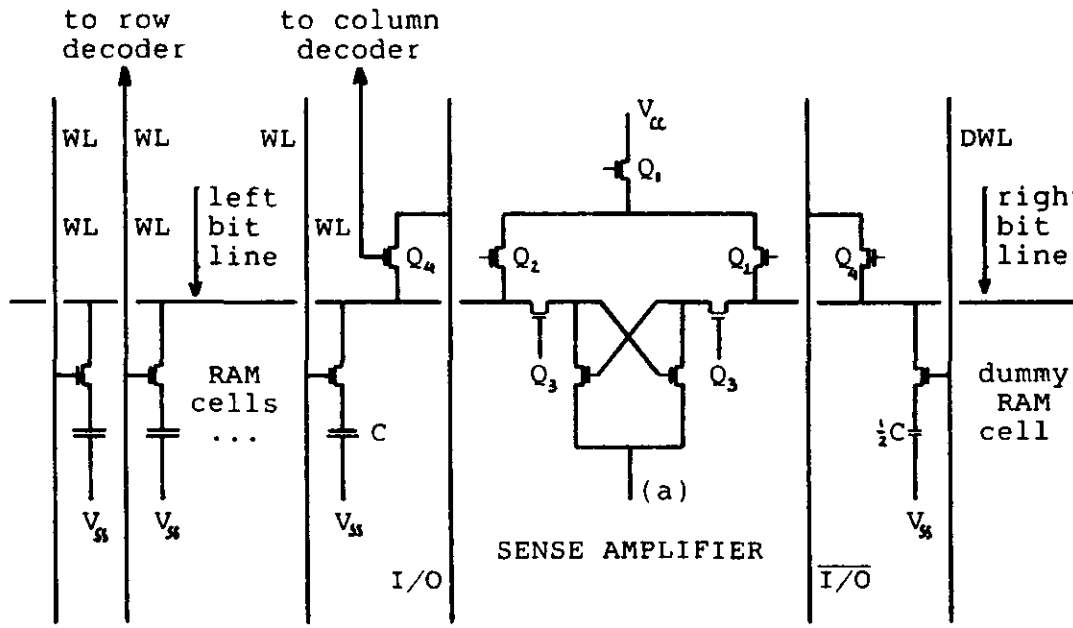


Fig.8 Operational scheme of a sense amplifier in a DRAM. The cells at the right side are not shown as well as the corresponding dummy cell at the left side.

A SRAM doesn't contain dummy cells because of the fact that both bit lines are directly connected to both sides of the sense amplifier during a read operation. According the scheme of figure 7.b, the column decoder acts as a multiplexer between the bit lines and the shared data bus. The last can be seen as an extension of the selected bit lines towards the sense amplifiers and the write drivers.

2.2.4 Control Circuitry

The descriptions above make clear that the internal timing and the sequence of the control signals play a very important part to avoid disturbances of the data streams. Above, the large dimensions and high component density require special measures for power supply and charge distribution.

2.3 Large Density RAMs

2.3.1 Developments

On the average every four years the density of semiconductor memories increases with a factor of four. Nowadays a very hard competitive struggle is going on about the production of the 4 Mega bit DRAMs and the 1 Mega bit SRAMs. The manufacturers who can deliver the first quantities may count upon considerable profits and have an opportunity to earn back their huge research efforts. However, one expects that in no time several producers embark upon the market which will be attended by a fast decrease of the prices.
In spite of these risks all prominent semiconductor manufacturers are strongly interested in the development of the new generation memory chips for two major reasons:

A. The memory segment of the semiconductor market grows rapidly due to the furthergoing integration of computer equipment and micro-processor applications in our society;
B. The required processing technologies are state of the art. If a manufacturer has overcome the associated production problems, then the elementary actions are under control to use these techniques straightforward for the processing of other device types.

2.3.2 Diagnostic Tools

The introduction of fundamental new production techniques demands for special measures. During both the R and D stage and the processing starting period, a lot of diagnostic tools are needed to get an insight in the wide variety of failure modes. Knowledge about the real fault origins makes it possible to optimize the process conditions as well as the design, resulting in better yield figures. One describes this stage with the term learning period [16],i.e. the ability to get an ever-increasing check on the process.

The newest generation memory chips however, have such high densities and huge complexity that even under stabilized conditions constant feed-back to the production environment will be necessary to achieve an acceptable yield. Statistical methods are used to analyse the large quantities of test results, offering the opportunity to trace irregularities of masks, process conditions, etc., in a very early stage.
Although the above mentioned measures, one foresees that even under stabilized conditions the final percentage all good devices will be less than 10% for the Mega bit memories. So, at least during the first years almost all the 256K, 1M and 4M DRAM designs as well as their comparable 64K, 256K and 1M SRAM counterparts have to be provided with

built-in redundancy by means of a few spare rows and columns in order to increase the yield. These spare rows and columns require only a few percent extra silicon area, whereas frequently occuring defects as single bit errors and isolated lines in the array can be repaired.

2.3.3 Built-in Redundancy

From a device under test a bit map is constructed indicating the topological locations of the fault places. After that a diagnostic examination will take place to determine whether the faults are limited to the restricted set of fault places that can be covered by the spare rows and columns. In case of a repairable device the on-chip redundancy is set in by blowing up some fusable links with a laser beam or electrical currents [17,18].
As long as the defective cells can be covered by the redundancy, the faulty rows and columns are disabled and replaced by the spare ones. The method requires programmable circuits to provide a flexible re-addressing mechanism. By using fusable links the address information of the faulty lines can be stored in a compare circuit. When the presented address matches with this information, the normal memory locations are deselected whereas an access takes place of the corresponding redundant cells. This technique will be explained using an example of a SRAM [18].

For each spare row a selection circuit is added as indicated in figure 9.a. The signal value of the spare row select line will always be low if all fuses are present. The address of a faulty row is programmed by blowing up the fuses of the complementary address lines. This causes that the select line goes high when an address value matches, hence resulting in a selection of that spare- row. Above, all other rows are disabled, including the faulty one, due to the dropped signal of the array deselect line.
Similar compare circuits are added to the column address lines. Since each normal address corresponds with two columns in this example, now the spare column as well as the faulty column pair are enabled together. To replace only one column, the defective one is switched off by blowing up the underlying fuse, whereas the fuse of the spare column is cut that serves the complement bit line pair, see figure 9.b.

Using such techniques one expects to win up to a factor of 30 during the learning period to get a yield of a few percent [19]. A disadvantage forms the decreased testability of these memories, these problems will not be attached in this thesis however.

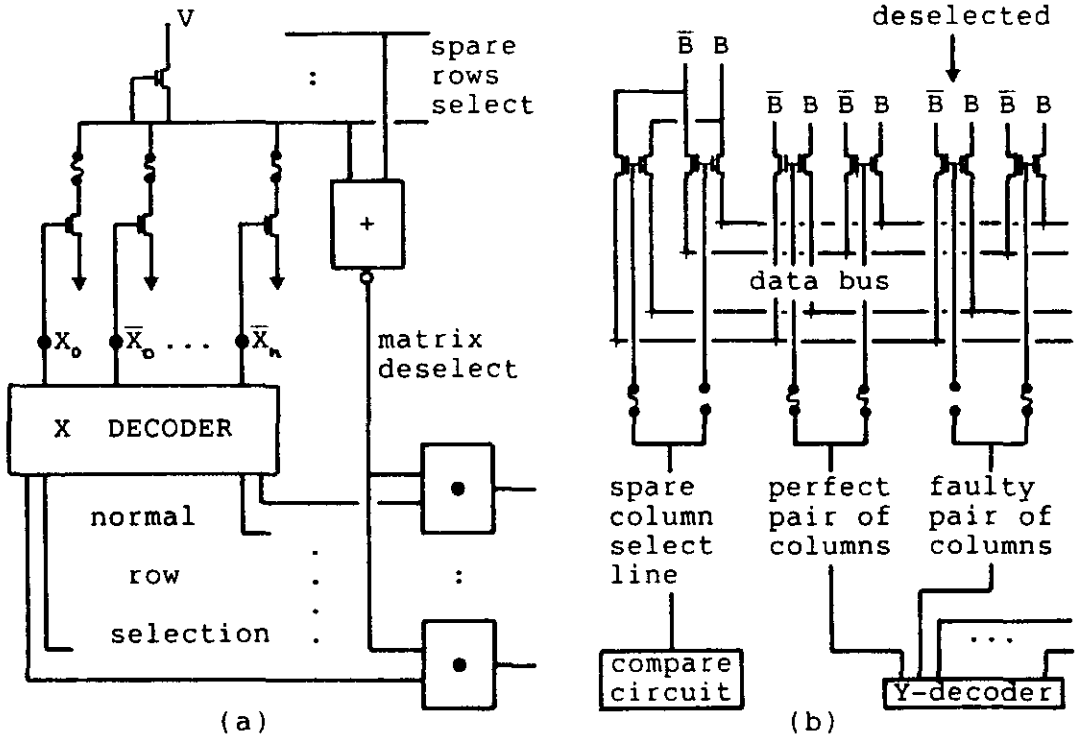


Fig.9 Additional circuits are added (a) to substitute spare rows for defective ones, and (b) to replace a single faulty column by the spare column.

In contrast, in the past several proposals were announced using on-chip error correcting facilities. The enormous increase of the critical memory area however, makes this approach completely impractical for the Mega bit memory range.

2.3.4 Address Scrambling

Memory designers are especially interested in a minimization of the critical silicon area to obtain better yield figures. For instance, considerable savings can be achieved by combining address lines of adjacent located memory cells. This results in a scrambling of the topological address sequence, i.e. successive logical address numbers are no longer related with neighbouring memory cells. Hence, tests for pattern-sensitive faults, see chapter 3, will be useless if one isn't familiar with the typical addressing scheme of the RAM device.
Memory test equipment normally cater for this problem with a descramble table [20]. The last can be programmed so that logical addresses are translated to topological addresses, see figure 10. However, the problem becomes more seriously

in case of a self-test implementation for embedded RAMs. This asks either for design rules to avoid arbitrary scrambling, or a test approach whereby the memory is considered as a black-box.

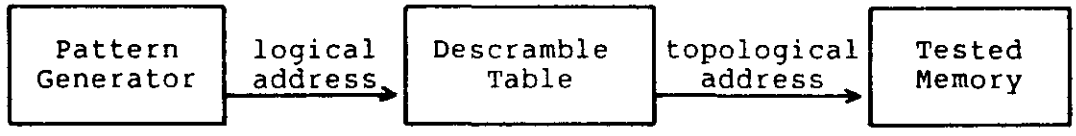


Fig.10 Conversion step of the address values to enable topological testing of the memory under test.

Besides address scrambling also data scrambling can be encountered, i.e. data inversions occur as function of the address values.

2.4 Embedded RAMs

Whereas all cells of a normal RAM device can be accessed in an easy way and simplify testing, large accessibility problems occur in case of embedded memories. The last are mainly used in logic devices as signal processors [21], single-chip microcomputers [22] and programmable peripheral controllers [23] to store temporary results. In spite of the fact that up to now these memories are relatively small, only a few thousands cells, their address and data busses are sometimes hard to control from outside. Although this difficulty can be solved by means of a scan chain, see section 1.4, the following disadvantages can be mentioned:

1. Addresses and data have to be shifted serially in and out the device. This requires a large amount of time due to the large number of test vectors and the slow shifting speed of the scan path;
2. The embedded memory can't be checked at its normal operation speed. Hence, typical time critical failures as slow recovery of the sense amplifiers, won't be detected.

A self-test implementation seems the obvious way to overcome these problems. We will discuss two self-test solutions in more detail at the end of the subsequent chapter.

3 MEMORY TESTING: A SURVEY

3.1 Introduction

From the beginning testing of RAM devices has been a difficult problem. Although the internal structure of the actual memory array is very regular, the wide variety of failure modes forms an important difficulty. Conventional testing methods based on the single stuck-at model, can't be successfully applied.

As a result of the steady growing memory dimensions, the number of test vectors to verify the logical operation of a memory, has become especially a matter of present interest. Besides, the rapid introduction of designs with embedded memories asks for a better understanding of the failure mechanisms.

From a historical point of view six major concepts can be distinguished for which test patterns were developed:

1. Ad hoc methods like "COLBAR", "DIAPAT", "Checkerboard", "Walking 0's/1's", "GALPAT", "GALTCOL", etc.[24,25]. Until now these tests are widely used in the industrial environment.
   However, these simple algorithms tend to be insufficient as only a few failure modes are covered. Efforts to construct more powerful procedures have led to very time consuming test algorithms without any guarantee of their completeness and effectiveness;

2. In 1977 Knaizuk and Hartmann presented an algorithm [24,26,27] covering all stuck-at faults in a RAM. A disadvantage of their idea was the assumption of a particular decoder structure. However, Nair [28] was able to construct a general applicable algorithm with a comparable number of memory operations.

3. At the same time Thatte [24,29] introduced a fault model based on the typical properties of a RAM device. Basically, his approach rests on the translation of faults in the address decoder and data transfer logic into equivalent faults in the memory array. This approach inspired several researchers to develop more efficient test algorithms based on the same assumptions;

4. From quite another point of view Hayes introduced the term neighborhood in 1980 [24,30]. On account of the physical device structure Hayes stated that the most probable fault effects have a limited range. No longer possible connections between arbitrary located cells were checked. Only the more plausible influences between adjacent memory cells are considered;

5. The next approach is based on the assumption of a fault model in combination with knowledge about the typical organization of the cells in the memory array [31]. Only testing for the most likely worst case situations leads to further optimized test sequences;

6. The last few years much effort has been undertaken to make memories more or less self-testable [32]. The proposals are based on the fault models mentioned above and promise to be very successful. Especially in case of embedded memories the advantages might be significant.

In the following paragraphs the principles listed above, are discussed in more detail. We thereby assume bit-oriented memory devices with a total number of N memory cells until further notice. This assumption makes it possible to weight the various algorithms mutually with regard to their complexity.
First an overview is given of the striking failure modes in RAM devices.

3.2 Failure Modes

3.2.1 Testing Methods

Three classes of tests can be distinguished [1,13]:

DC parametric, aspects like output voltage, rise and fall times of the signals, power consumption, fanout capabilities, etc., are verified in this mode. However, the values and margins depend strongly on each specific state;

AC parametric, at the circuit's operational speed magnitudes as access, refresh, recovery, set-up and hold times are checked;

Functional, detection of permanent failures, i.e. testing the internal device structure for its logical behaviour. This envolves the examination of the memory cells for stuck-at faults, the working of the decoder, etc., using sophisticated data patterns with relaxed timing conditions.

In general DC parametric faults can be visualized as a very special type of failure mode. The detection offers specific tests depending on the typical implementation and processing techniques of the considered device. An universal approach is unfortunately impossible, so each device type demands its own solution. In view of these facts no further attention will be given to DC parametric testing.

Although this report also passes the problem of AC parametric testing, functional test procedures detect many AC malfunctions when the tests are applied to the memory at the maximum operational rate.

Functional testing is applied to detect permanent faults and can be done with already known test procedures based on general fault models. These faults don't depend as much on the particular situation as the DC faults and some AC malfunctions do. An investigation of the most likely functional failure modes makes it possible to sensitize the test procedures to these faults, eventually combined with knowledge about the internal structure of the considered RAM type. For instance, large RAMs are more likely to suffer from the effect of parasitic capacitance coupling on adjacent memory cells than the smaller types do. Taking these properties into account, one might ultimately provide a good fault coverage with a small number of test vectors.

In addition, the combination of AC and functional tests, i.e. functional test patterns performed with a shortened cycle time to detect time-dependent faults as well, should give considerable savings in test length. Especially self-test methods are promising in this sense, because they make functional testing possible at the normal operational speed.

3.2.2 Typical Malfunctions in a RAM

A typical block diagram of a semiconductor RAM was given in figure 6 of section 2.2. One or more address registers, address decoders, a memory cell array, read/write logic and a data register can be seen as the main functional parts. At this point we suffice with an enumeration of the more general functional malfunctions mentioned in literature [1,13].

1. Memory cell array (functional);
2. Malfunctions of the address decoder (functional);
3. Read/write logic (functional);
4. Write recovery time (AC);
5. Sense amplifier sensitivity (AC);
6. Sleeping sickness (AC DRAM) cq. data retention (AC SRAM).

ad.1. Stuck-at faults:
the content of an individual cell always has the same logical value;

Coupling faults:
forcing a write or read operation in a cell also influences the content of one or more arbitrary located other cells, and mutual influences between cells due to parasitic effects;

Pattern-sensitive faults:
coupling faults which are restricted to neighbouring cells;

Shorts and opens of word and bit lines:
these faults either result into multiple accesses, data mutilation or inaccessible strings of cells.

ad.2. One or more memory cells can't be accessed or multiple cells are accessed concurrently. This failure manifests itself in the impossibility to store data into some regions of the memory, or in the affection of data in non-addressed cells due to memory operations at some other cells.

ad.3. Shorts, opens and stuck-at faults can be present in the data transfer logic, resulting in data mutilation.

ad.4. In the presence of write recovery failures the memory can't deliver the correct information within the desired access time when a write cycle is immediately followed by a read operation. Although write recovery is an AC parameter, malfunctions have to be detected with functional test patterns because of the data sensitiveness.

ad.5. When a number of operations have taken place with equal data, a sudden change of the considered data values might lead to transition problems in the sense amplifiers. This failure type also belongs to the AC parametric class, but the faults can be detected by functional patterns as well.

ad.6. When in a DRAM the stored information expires within the hold time, this malfunction is indicated with the term sleeping sickness. The effect is a result of a too rapid fall of the capacitor charge before a refresh operation has taken place. SRAMs can also suffer from loss of information due to failures in the cells, called data retention. The presence of sleeping sickness or data retention can only be demonstrated with data patterns in combination with wait cycli between the successive memory operations.

This mixture of failure mechanisms have led to the six test approaches mentioned in section 3.1. They will be discussed now one by one.

3.3 Ad-hoc Test Procedures

3.3.1 Historical Background

With the introduction of RAMs a wide variety of data patterns have emerged [1,24,25,33]. Usually a new test pattern was introduced when difficulties appeared with the detection of a particular failure mode. They were added to already existing test procedures to get a better fault coverage. For instance, the sleeeping sickness problem in DRAMs became better traceble with patterns like "COLBAR", "Checkerboard" and "Refresh".
At the beginning stage of the RAM development this was the only manner to tackle the testing problem. On the one hand, sufficient knowledge about the typical RAM failures was lacking, especially from the memory user point of view. On the other hand, one was searching for a general applicable fault model. But the devices were relative small, so test time considerations didn't play an important part at that time, even if the number of memory operations was proportional to N**2 (N square). Problems in this sense were solved by the introduction of more ingenious test equipment.

3.3.2 Test Time Considerations and Fault Coverage

The rapid growth of memory sizes and quantities demanded for test time limitations. Table 1 gives an indication of the required test time for various pattern complexities as function of the device size. Especially procedures like "GALPAT" (N**2) become impracticable for memory dimensions of 16K or larger.
In turn, increasing device densities and speed resulted in a rise of failure modes. Apart from the specific fault sensitiveness of the developed ad hoc procedures, their overall fault coverage is questionable. They are heuristic in nature, thus practical experiments have to be employed

	N	N.logN	N**3/2	N**2
1 K	$1.0\ 10^{-4}$	$1.0\ 10^{-3}$	$3.3\ 10^{-3}$	$1.0\ 10^{-1}$
2 K	$2.0\ 10^{-4}$	$2.2\ 10^{-3}$	$9.3\ 10^{-3}$	$4.2\ 10^{-1}$
4 K	$4.1\ 10^{-4}$	$4.9\ 10^{-3}$	$2.6\ 10^{-2}$	1.7
16 K	$1.6\ 10^{-3}$	$2.3\ 10^{-2}$	$2.1\ 10^{-1}$	$2.7\ 10^{1}$
64 K	$6.6\ 10^{-3}$	$1.0\ 10^{-1}$	1.7	$4.1\ 10^{2}$
256 K	$2.6\ 10^{-2}$	$4.7\ 10^{-1}$	$1.3\ 10^{1}$	$6.6\ 10^{3}$ = 1.8hrs
1 M	$1.0\ 10^{-1}$	2.1	$1.1\ 10^{2}$	$1.1\ 10^{5}$ = 30.6hrs
4 M	$4.2\ 10^{-1}$	9.2	$8.6\ 10^{2}$	$1.8\ 10^{6}$ =500 hrs

Table 1 Test time in s as function of the pattern complexity vs. address range (N=1K,2K,...,4M), assuming a cycle time of 100ns.

for the determination of their detection capabilities. But even then one has no insight in the efficiency of the considered patterns and hard evidences about their fault coverage are still missing.

3.3.3 Engineering Area

The arguments mentioned above make clear that the ad hoc approaches became impractical in the production environment. So, manufacturers were forced to look for better and powerful solutions. Nevertheless, most of the patterns mentioned in this context are still used in the engineering area [33]. New designs and testing off-line require worst case examinations. Because of their specific fault sensitiveness the ad hoc patterns are appropriate for this purpose.

Table 2 of section 3.8 contains the most well-known test procedures of this sort arranged in order of their complexity and specific fault coverage. See Appendix A for a description of these algorithms.

3.4 Stuck-at Faults in RAMs

When all stuck-at faults are restricted to the memory array at least 4N operations (bit wide device) are necessary to investigate each memory cell. First, we have to initialize all the N cells to the 0-value. Secondly, all cells have to be read examining if one or more are stuck-at-1. Thirdly, the whole memory is filled with 1's. At last, all cells are read to detect stuck-at-0 faults.

THE "MSCAN" TEST

The ad hoc procedure "MSCAN" (Memory Scan) would satisfy the above conditions. However, if a faulty decoder maps each address consequently to only one memory location, just that particular cell will be examined [24]. So, "MSCAN" does not come up to the expectations.

THE "MARCHING 0's/1's" TEST

In advance the ad hoc procedure "Marching 0's/1's" was developed requiring 14N operations. Although this pattern guarantees that all cells can be accessed and tested for stuck-ats, the procedure is far from efficient. The provable fault coverage agrees with that of the "MATS+" algorithm [24], discussed further on in this section.

THE "ATS" PROCEDURES

Knaizuk and Hartmann presented in 1977 [26] the algorithm "ATS" and proved that it would detect all single stuck-at faults in a RAM with only 4N memory operations. In a second paper [27] published in the same year, they demonstrated that "ATS" even covers all multiple stuck-at faults. A disadvantage was the condition that the decoder is constructed of simple AND and NOR gates with no reconverging paths, a non-creative design.
Above, the addresses were partioned into three categories, by which alternately memory operations are executed at cells belonging to one category. This results in a troublesome address sequence for self-test implementation, because three is no power of two.

Bardell and McAnney enlarged this algoritm with a preliminary initialization step to detect also open bit-rail faults [34]. For instance, if the failure makes it impossible to write 1 to the cell and the cell happens to pull to the logical 1 after power-on, an open write 1 bit-rail fault is undetectable by the ATS operation sequence write 1, read 1, write 0, read 0. The detection of this fault is provided by the inclusion of a priliminary write 0 operation. They stated that the "MATS" procedures may be improved in the same way.

THE "MATS" PROCEDURES

In 1979 Nair [28] showed that by means of a rearrangement of the address sequence the problem of the difficult address order of the "ATS" procedure could be solved. The modification leads to simple march patterns, which makes implementation a very simple task. The solutions are called "MATS-and" and "MATS-or", and take again the optimal number of 4N memory accesses if the decoder has a wired-and or wired-or behaviour, respectively. I.e., in case of a wired-and construction one or more stuck-at-0 faults may be detected in a field of all 1's with only one test, whereas similarly one or more stuck-at-1 faults are detectable in a field of all 0's for a wired-and behaviour.
When the decoder structure is unknown, the "MATS+" algorithm, which requires 5N-2 memory operations, will detect also all stuck-at faults in a RAM. Both achievements provide that the "MATS" procedures are very suitable for a self-test application, see for instance [35].

Unfortunately, the stuck-ats form a limited part of the many failure mechanisms in RAMs, hence the reach of these algorithms is relatively small. The procedures are included in table 2 of section 3.8 and are described in Appendix A.

3.5 Testing for 2-Coupling Faults

3.5.1 Fault Model for 2-Coupling Faults

In a paper of Thatte and Abraham [29] a fault model was introduced based on a sub-division of a RAM into three functional blocks. They distinguish the memory cell array, the decoder logic and a third part consisting of the sense amplifiers and write drivers. The model is fixed on the detection of permanent functional faults in a RAM. Very efficient algorithms were achieved as result of the translation of failures in the addressing and data circuits into equivalent faults in the memory array.

If the decoder doesn't access the addressed cell, the fault can be viewed as a stuck-at fault in the unattainable memory location, whereas multiple accessed cells can be associated with coupling faults between cells in the memory cell array. In addition, stuck-ats of the output and write drivers are similar to stuck-ats in the corresponding memory regions. Likewise, shorts between data lines can be seen as coupling faults in the memory cell array.
Therefore, Abraham and Thatte stated that testing the array for stuck-ats and 2-coupling faults makes seperate tests of the decoder and read/write logic superfluous.
In this context a parallel can be drawn to the single stuck-at model versus the multiple stuck-at model. If two memory cells are mutually coupled, this fault may remain undetectable due to possible couplings of these cells with one or k other cells, called 3-coupling or k-coupling respectively. Abraham and Thatte assumed that a test pattern which can detect each single 2-coupling fault will detect the majority of the multiple couplings as well.

Before discussing the accessory algorithms in more detail, we will pay attention to the comparable ad hoc procedures.

3.5.2 Ad-hoc Test Procedures

THE "COLBAR" TEST

First, the "COLBAR" (Column Bars) test can be seen as a simple data pattern to detect coupling faults between cells in adjacent columns. The columns are written with alternating logical values after which all cells are read. The process will be repeated with complementary data. Although the complexity of the pattern is merely 4N, the fault coverage is very bad [24]. Neither most of the stuck-ats, nor coupling faults between cells in the same column as well as coupled cells in columns at intervals of twofolds are detected. Above, a faulty decoder curtails the detection range significantly, comparable with the effects of the "MSCAN" procedure, mentioned in the previous section.

THE "MARCHING 0's/1's" TEST

Secondly, again the "Marching 0's/1's" test can be quoted in this context. In accordance with the earlier observations, this procedure detects all stuck-ats in the memory array. The coverage of 2-coupling faults is limited [24], however. If for instance a 0-to-1 (1-to-0) transition in cell i forces a write of a 0 (1) in cell j (i<j), the failure won't be detected [24].

"GALPAT" AND OTHERS

Thirdly, the "GALPAT", "Walking 0's/1's", "GALTCOL" and "GALTROW" are famous with regard to their detection capabilities for address transition failures. Although the squared procedure "GALPAT" covers all stuck-at faults and a huge part of the 2-coupling faults in the memory array [24], its enormous number of operations makes the procedure fully unsuitable for the modern RAM dimensions.
The properties of the "Walking 0's/1's" algorithm are even worser. Small improvements with respect to its complexity has resulted in a significant decreased fault coverage because the address transitions aren't attended anymore with complementary data.

The "GALTCOL" and "GALTROW" procedures can be visualized as impressive simplifications of "GALPAT" to overcome the lengthy test time requirements. Notwithstanding these improvements, the declined test area to cells in the same column or row, is attended by an important decrease of the fault coverage. The detectable coupling faults are almost completely restricted to cells in the same row or column, depending on the used test pattern. Besides, the time savings are temporary becauce of the growing memory dimensions, see table 1.

Though test engineers will put forward that algorithms like "GALPAT" are very effective, the arguments given above have to be visualized in the context of the detection capabilities for permanent faults. Their pattern complexity doesn't counter-balances to the achievements. Therefore we now address the more efficient procedures proposed by Thatte and Abraham, as well as improvements on their ideas published in following papers.

3.5.3 Tests for Coupling Faults

THATTE AND ABRAHAM's $\{8N^2 \log N\}$ TEST

In [29] Thatte and Abraham postulated their requirements for functional testing based on the fault model described at the begin of this section. Every pair of memory cells should

undergo the following state transitions:

1. force a transition from 0 to 1 in cell i, when cell j stores a 0 as well as a 1;
2. the same in case the content of cell i changes from 1 to 0;
3. repeat both steps after exchanging the roles of cells i and j.

These claims guarantee that every cell which is stuck-at or which fails to suffer a 0-1-0 or a 1-0-1 transition, as well as 2-coupling faults between any arbitrary cell pairs will be detected, satisfying the conditions of the fault model.
The in [29] proposed test pattern has a complexity of $8N.^{\frac{1}{2}}\log N$, but the address sequence is rather difficult as result of the partitioning of the memory into three categories.

30N and $\{N + 32N.^{\frac{1}{2}}\log N\}$ TESTS of Nair et.al.

One year later Nair, Thatte and Abraham [36] came with a modified version based on the same fault model, called "Test A", taking only 30N operations. Besides the decreased complexity, the procedure is much easier qua implementation due to its simple marching address structure.
The paper also contains an extended algorithm "Test B" of complexity $\{N + 32N.^{\frac{1}{2}}\log N\}$. This algorithm has the same fault coverage as the 30N procedure including restricted 3-coupling faults, i.e. satisfying the condition that coupling faults between two cells may not be detected due to transitions in a third cell. A disadvantage forms again the classification of the memory cells, making the implementation of the algorithm more difficult. Above, the extention to 3-coupling faults involves a dramatic increase of the complexity. One has to ask oneself whether this enlargement is practical significant, because of the various other failure mechanisms in a RAM and the little improvement in fault coverage.

SUK and REDDY's AND MARINESCU's 11N,15N,17N TESTS

In addition, several algorithms based on the same assumptions were announced by Suk and Reddy [37] and by Marinescu [38]. The number of memory operations varies in these proposals from 11N up to 17N.
Suk and Reddy proved that their "Test A", a marching test sequence of length 14N, will detect a large part of the coupling faults between any arbitrary cell pairs, transition, stuck-at and multiple access faults on the condition that only one type of fault at a time is present in the RAM under test. The slightly different "Test B", complexity 16N, should give almost the same fault coverage

as the 30N algorithm of Nair et.al. if the memory doesn't suffer from multiple accesses. The evidence of "Test B" is lacking in their paper, although they are referring to a desertation of Suk.
The content of the paper [38] presented by Marinescu has a more mathematical back-ground. "Algorithm A", complexity 15N, is identical to Suk and Reddy's "Test A", except for the argumented ommission of an initialization of the memory in an all zero state by Suk and Reddy. "Algoritm B", complexity 17N, slightly differs from Suk and Reddy's "Test B", whereas Marinescu's "Algorithm C" of length 11N has a questionable fault coverage due to further restrictions.

PAPACHRISTOU AND SAHGAL's 37N and {37N + 24N.²logN} TESTS

In 1985 Papachristou and Sahgal [39] came with new ideas for the 30N and {N + 32N.²logN} algorithms of Nair et.al.[36]. They connect the 30N one with the "Test A" and "Test B" procedures presented by Suk and Reddy [37]. The paper examines the problem of the 2-coupling faults in a very extensive and fundamental way. Their investigations lead to a marching test pattern of length 36N. The complexity ought to be 37N, on account of the fact that in the algorithm a necessary initialization step of the memory in an all zero state has been left out.
A detailed comparison of the 30N procedure and this 37N pattern results in the conviction that the differences are negligible. The 30N one is a shorter due to the repetitive changing of the addressing order from increasing to decreasing address numbers, an effective way to detect 2-coupling faults, whereas the addressing order of the 37N algorithm is only altered in the middle.
Papachristou and Sahgal also developed an additional procedure of complexity {24N.²logN} to detect the same class of restricted 3-coupling faults as the {N + 32N.²logN} one proposed by Nair et.al. did. The savings in test time are small and the address generation has the same problems mentioned earlier.

3.5.4 Considerations

The conclusion can be made that about 30N memory operations are enough to check a RAM on permanent transition, stuck-at, multiple access and 2-coupling faults, even in case of the simultaneous occurence of all these functional faults. Which test pattern finally will be chosen remains a question which depends of the desired fault coverage and implementation structure. After all, as the main result it can be stated that reliable and simple march patterns have become available with a linear complexity instead of their squared counterparts.
An extension to restricted 3-coupling faults involves a

considerable rise in test time and difficult addressing sequences, opposite to a small increased fault coverage.

The discussed procedures are part of table 2 of section 3.8 and descriptions of the algorithms are given in Appendix A.

3.6 Pattern-Sensitive Faults

3.6.1 Historical Background

Parallel to the 2-coupling fault approach another fault model has been introduced. The first article was published in 1975 by Hayes [40]. He makes a comparison between a memory and a sequential Mealy machine, see Appendix B. The idea is based on the assumption that most of the malfunctions in large density RAMs come on the account of pattern-sensitive influences between memory cells. An exhaustive test for all possible data patterns and interactions between arbitrary memory locations leads to the unrealistic checking sequence of $(3N^2 + 2N).2^N$ operations, in spite of the use of Eulerians paths (see Appendix B) to minimize the number of write actions. This enormous complexity arises from the fact that in essence all possible coupling faults in the memory are considered. Although the drawed parallel between a RAM and a Mealy machine turned out to be impractical, the pure theoretical approach appeared useful in formulating the necessary test elements for this class of faults.

3.6.2 Neighborhood

Although Hayes [40] stated that it would be sensible to examine only influences between adjacent memory cells, he didn't succeed in transforming his ideas into a practical solution. Scrini [41] tackled this problem from a more practical point of view. He made some constraints about the reach of these influences. The main goal of a test for pattern-sensitive faults is that it will detect extreme charge leakages of storage cells and the effect of parasitic capacitance coupling on storage cells. So, to the opinion of Scrini it should be sufficient to investigate only the influences between direct neighboring cells. Therefore, he adopts the star-shaped 5-cell configuration as neighborhood, see figure 11a. Although the 9-cell square neighborhood has been mentioned as a slightly better alternative, see figure 11c, all the papers on this topic only consider the 5-cell neighborhood seriously to avoid unmanageble test patterns.

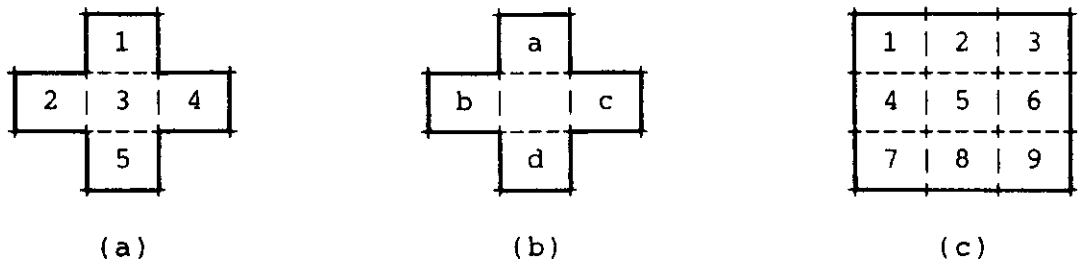


Fig.11 Neighborhood types; whereby (a) stands for the closed and (b) for the open 5-cell neighborhood whereas (c) reflects the 9-cell configuration.

However, the constraint assumes a strict relationship between logical and topological addresses, i.e. a coherence between address numbers and adjacent memory locations. It depends on the considered device type in which way the addresses are scrambled as result of the particular implementation of the decoders, see section 2.3.4. It is remarkable that this aspect has been neglected in the literature. For the moment we assume that this condition has been fulfilled.

3.6.3 Tiling of the Memory Array

In a next paper [30] Hayes proposed a formal description of Scrini's ideas by a tiling of the array with nonoverlapping or closed neighborhoods, see figure 11a. The typical arrangement of the cells repeats itself with squares of 5x5 cells, see figure 12. The cells are labeled in such a way that each cell is surrounded by exactly four cells of the other categories.

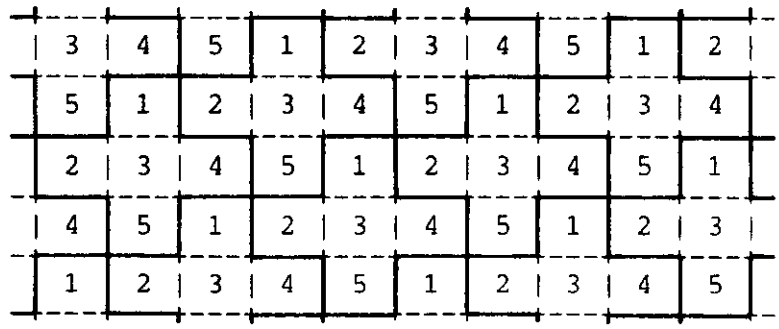


Figure 12
Tiling of the memory array with the closed neighborhoods.

The other authors [42,43] use the so called overlapping, deleted or open neighborhoods, see figure 11b. In that case the test or base cell is deleted from the set of cells forming the neighborhood. Besides the four categories A, B, C and D, Suk and Reddy presented a furthergoing division of the cells into two modes. The test cells in mode 1 acts during mode 2 as surrounding cell, vice versa, see figure 13

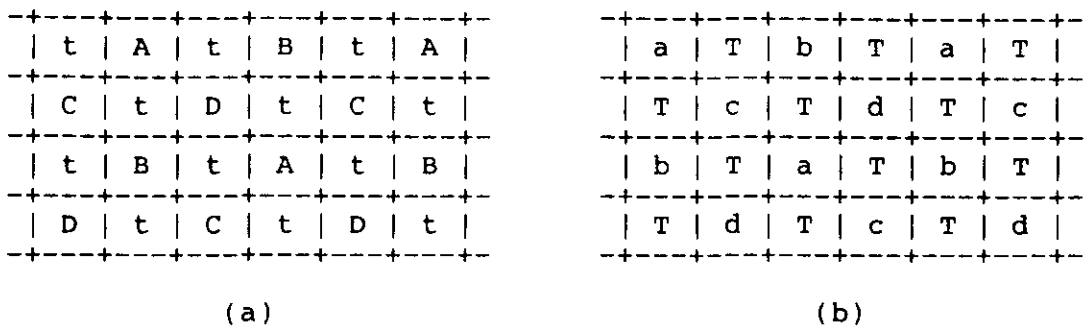


Fig.13 Tiling of the memory array with deleted neighborhoods the test cells in situation (a) are neighborhood cell in situation (b), and vice versa.

A major problem associated with the tiling and labeling methods forms the increased complexity of the address generation. Especially self-test applications will give difficulties, although the method using deleted neighborhoods will be easier to implement due to the fact that the distances between the cells are always a power of two.

3.6.4 Types of Pattern-Sensitive Faults

In the papers on this topic slightly different terminologies has been used. We will only handle the unambiguous definitions proposed by Saluja and Kinoshita [43]. They distinguish static and dynamic adjacent pattern-sensitive faults. The first type concerns the situation that the content of the base cell is affected by the contents of the surrounding cells belonging to the same neighborhood. The dynamic fault type occurs when the content of the base cell is influenced by transitions in the associated neighborhood cells. Moreover, Suk and Reddy [42] consider in addition the situation that the content of the base cell can't be changed due to certain data patterns in the cells of its deleted neighborhood. The general approach of Hayes [40] considers both the above types and the non-transition write and read operations.

3.6.5 Test Procedures

The first procedure was presented by Scrini [41], who found by trial and error a state sequence of 32N write and 32N read operations. Each base cell is set to the 0-value and all the 16 different data patterns in its deleted neighborhood are sequenced. The procedure is repeated with the 1-value and the verification of the base cell for each situation guarantees that all static pattern-sensitive

faults are covered.
Although Hayes proved (using Eulerian paths, see Appendix B) that the transition write sequence would take 32N operations, Saluja and Kinoshita [43] pointed out, by using Hamiltonian paths (see Appendix B), that even a lower bound of 31/5N write actions will satisfy the demands to perform a optimal state sequence for the static pattern-sensitive faults. They showed that a single write operation can effect the state of exactly 5 cells, and in combination with the tiling configuration of figure 13, the number of writes could be reduced to 8.7/8N writes by forcing only transitions in one set of cells at a time. Unfortunately, still 32N read operations were needed since no minimization of the read actions was implied in their algorithm.

Suk and Reddy [42] assumed the more comprehensive fault class of the dynamic pattern-sensitive faults. This approach requires the complcated Eulerian paths (see Appendix B) to achieve an optimal write sequence. The rather difficult test procedure TANPSF1 takes about 100N memory operations to detect these dynamic faults, despite the application of similar saving methods used by Saluja and Kinoshita. In addition, about 65N actions are needed to discover cells which content can't be changed as result of data patterns in the cells of their deleted neighborhood. The practical importance of the latest fault type is questionable, however.

The model defined by Hayes [40] incorporates all static and dynamic faults as well as faults due to non-transition operations. Hayes calculates that such a test pattern will take at least 544N memory accesses [30] for the 5-cell neighborhood. Although a reduction might be possible using the methods proposed by Suk and Reddy, the model seems too general.

3.6.6 Considerations

When the internal addressing mechanism of a RAM is known, testing for pattern-sensitive faults becomes possible. Only the most probable influences between adjacent memory locations can be considered to avoid infeasable test procedures. However, programs which cover both static and dynamic failures tend to have considerable test length and the generation of their address sequence gives difficulties. Hence, most of the attention in literature has been spent on the derivation of simplified algorithms which detect merely static pattern-sensitive faults. A practical and efficient solution was proposed by Saluja and Kinoshita, requiring about 40N memory accesses.
In general, the implementation of these tests for pattern-sensitive faults won't be easy. Especially the labeling methods require difficult address assignments.

The discussed procedures are taken up in table 2 of section 3.8. Descriptions of the algorithms are given in Appendix A, whereas Appendix B contains explanations of the terms Mealy machine, Hamiltonian path and Eulerian path.

3.7 Structural Testing

The algorithms discussed until now were all based on a certain degree of mathematical fault modelling. Although remarkable reductions could be established, as seen by comparing the algorithms of section 3.3 with their improved counterparts in section 3.4, 3.5 and 3.6, the internal structure of the memories is hardly used in the reduction efforts. It is true that Thatte visualizes a partition between the memory cell array, the address decoders and the read/write logic, but furthergoing attempts are lacking. Testing for pattern-sensitive faults assumes a strict relationship between the actual and the topological addresses, but the connection ceases at this point.

One can foresee however, that new substantial savings in test length are only possible when the typical behaviour and structure of the memory circuit is involved in the test strategy. The approach takes a correlation between fault origins and the internal memory structure into account. This way of reasoning has been explained in two papers [31,44] published recently. The fundamental difference with regard to the previous proposals is the way the theoretical fault models are related to physical failures. You and Hayes presented a table [44] which contains the most frequently occuring failure modes and their test requirements. These requirements are reported in terms of the desired operation sequence and the smallest set of cells which has to be investigated.

You and Hayes developed a minimal test sequence by striking overlaps between sequences covering different failure modes. In combination with their self-test scheme an algorithm was presented with complexity N**1/2. A promising achievement, mainly obtained by small adaptations of the control circuitry and the sense amplifiers making it possible to test several cells and array partitions in parallel.
Looking at the resulting fault coverage, their approach is even more promising. You and Hayes stated that the typical malfunctions in a RAM, see section 3.2, will be detected by their algorithm. However, a comparison with the previous discussed procedures is difficult to make, due to the fact that the proposals of You and Hayes are based on both tests for worst case situations and the typical implementation of a RAM. For instance, their test for pattern-sensitive faults is performed with a neighborhood consisting of the

cells immediate surrounding the base cell, cells sharing the same bit line and cells opposite to the sense amplifier [44], see figure 14. The accessory worst case data pattern also performs a test for sleeping sickness and bit-line imbalance. These considerations are related to the typical structure of the DRAM discussed by You and Hayes.

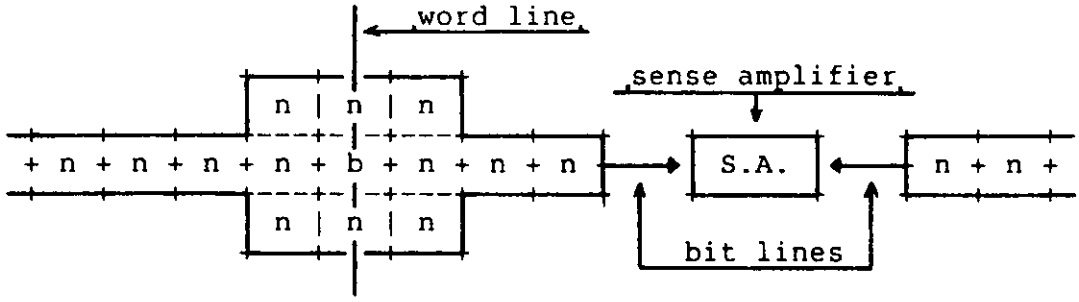


Fig.14 Neigborhood cells "n" belonging to the base cell "b" according to the opinion of You and Hayes.

A disadvantage of the proposal concerns the additional chip area needed for the self-test implementation. Jou and Hayes calculated that the duplicated peripheral circuit and the on-chip test generator will require a 26% increase in silicon area for a 64K DRAM. However, one can also consider a solution whereby the test is still performed with an off-chip test generator to overcome the disadvantage largely, whereas the decreased test time can be retained.

3.8 Review of the Test Procedures

We will summarize the previous sections by summing up the main elements encountered until now. In table 2 most of the well-known test procedures are presented. The table is devided into three sections, each containing the algorithms with similar fault coverage properties. This classification in order of the fault coverage is perhaps little obscure. It has to be regarded as a first attempt to bring some clearness in the wide variety of test procedures. Although the detection properties for permanent faults can be proven, the real effectiveness of a test procedure depends highly on several vague circumstances. On this point, neither comparitive figures nor well surveyable papers have been published in the last years. Several reasons can be given this. First, manufacturers do not release useful information in view of the violent market competition. Secondly, the drafters of a new test procedure suffice normally with a theoretical prove of correctness, whereas practical comparisions with existing methods are missing.

procedure	#operations	fault coverage
A. Stuck-at faults (section 3.4)		
1. MSCAN [24]	4N	stuck-ats if the address logic is fault free
2. ATS [26,27]	4N or 13/3N	all stuck-ats and check of the decoder, if the decoder has a non-creative design
3. Marching 0/1's [24]	14N	all stuck-ats, decoder check and some 2-coupling faults
4. MATS [28]	4N or 5N	all stuck-ats and decoder check
B. 2-Coupling faults (section 3.5)		
5. COLBAR [24]	4N	stuck-ats and some 2-coupling faults if the address logic is fault free
6. MASEST [25]	10N	improvement over COLBAR to detect even some AC failures
7. Marching 0/1's [24]	14N	all stuck-ats, decoder check and some 2-coupling faults
8. Shifted Diagonals [1]	$6N + 2N^{3/2}$	especially suited to detect sense amplifier sensitivity failures, also stuck-ats and some couplings
9. Algorithm C [38]	11N	stuck-ats and a large part of the 2-coupling faults, only if one type is present at a time
10,11. Test A [37] or Algorithm A [38]	14N or 15N	stuck-ats and a huge part of the 2-coupling faults, only if one type is present at a time
12,13. Test B [37] or Algorithm B [38]	16N or 17N	stuck-ats and a huge part of the 2-coupling faults, if no multiple accesses occur
14. Walking 0/1's [25]	$2N^2 + 8N$	stuck-ats and a large part of the 2-coupling faults
15. GALPAT [1]	$4N^2 + 4N$	stuck-ats, slow recovery faults in the sense amplifiers and almost all 2-coupling failures
16. GALTCOL [25] or GALTROW	$4N^{3/2} + 4N$	same as GALPAT restricted to the cells in the same column or row
17. Thatte&Abraham [29]	$8N.^2\log N$	stuck-ats and all 2-couplings
18,19. Test A [36] or Papachr.&Sahgal [39]	30N or 37N	same fault types as 17.
20,21. Test B [36] or Papachr.&Sahgal [39]	$N+32N.^2\log N$ or $37N+24N.^2\log N$	same as 17 and also some restricted 3-coupling faults
C. Pattern-sensitive faults (5-cell neighborhood, see section 3.6)		
22. Surround Disturb	24N	questionable PSFs fault coverage
23. API test 1 [41]	64N	single static pattern-sensitive
24. API test 2 [43]	41N	" " " "
25. Proc.TANPSF1 [42]	99.5N	a large part of the single dynamic pattern-sensitive faults
26. Hayes [30]	544N	single static and dynamic pattern-sensitive faults as well as PSFs due to non-transition operations

Table 2. The previous discussed algorithms listed in order of their detection properties and complexity.

Thirdly, the effect of a test also depends on the considered RAM type as result of its specific internal structure and behaviour.

After this extensive survey of the recent literature on this topic, the following conclusions can be made:

a. The assumption of a fault model is useful in constructing optimal test sequences;

b. Minimal test algorithms are available to check a RAM for stuck-ats and other permanent faults;

c. Little progress can be viewed with respect to the testing for pattern-sensitive faults;

d. Promising results can be reported whereby fault models are transferred to physical failures in the internal circuitry;

e. The detection of AC failures remains an unsolved problem and was scarcely addressed.

3.9 Self-Test and Embedded RAMs

In the last few years a wide variety of papers has been published presenting self-test techniques for memories [32,34,35,44,45,46,47]. The paper of You and Hayes [44] was already mentioned in section 3.7, the proposals in the remaining articles are based on the fault models for stuck-at and 2-coupling faults.

3.9.1 Large Density Memories

Processing the large density memories of today is coupled with the introduction of fundamental new processing conditions. To achieve a moderate yield during the learning period, all the attention is addressed to diagnostic measures. Also, the minimization of the silicon area forms an important economic consideration. So, at the beginning self-test proposals won't make a change due to the extra chip area and the decreased observability of the failure origins. Above, the introduction of repairable devices have changed the test strategy and hinders the introduction of self-test.
Hence, the trade-off between more sophisticated test equipment and extra chip area for self-testing purposes will be more realistic at a later stage, i.e. when most of the design errors and processing shortcomings have been conquered. See also section 2.3.

Therefore, the self-test proposal of You and Hayes [44] and the design for testability approach of Saluja and Thang Le [46] have to be placed in this context. The last method concerns an extension of the address decoders with additional circuitry enabling the possibility to perform plural read and write operations simultaneously. With about 25N memory accesses all stuck-ats, static adjacent pattern-sensitive faults and decoder faults can be covered. Although the authors didn't give figures with regard to the increase in silicon area and speed performance, they estimate only minimal changes.

3.9.2 Embedded RAMs

In chapter 1 the trend in modern IC design was addressed whereby more and more small RAM circuits can be found on the same chip next to modules as PLAs and ROMs. In consequence, test engineers deal with the increasing problem of the observability and controllability of these memory parts due to the surrounding circuitry. Self-test of the embedded memory could offer a solution for this problem.
Notacing the large number of operations for a memory test, it seems almost impossible to perform a complete test from outside using the traditional serial scan. Apart from these test techniques, the memory can't be checked at its operational speed, so the fault coverage will remain restricted to the permanent faults.
Hence, much emphasis has been addressed to make embedded RAMs self-testable in a convenient way. Additional hardware is used to generate the very specific set of RAM test patterns. The methods presented in literature [32,35,47,48] all make use of address and data generators controlled by a small PLA or ROM. Only a compact test scheme is needed by which the shift registers are programmed to check the memory with simple march patterns.

Sun and Wang [35] adopted the "MATS+" algorithm of Nair [28] to make an embedded RAM self-testable. An additional scan technique was used to check the test circuits and data lines for stuck-ats and bridging faults. Similarly, Varma et al. [32] implemented the 30N procedure of Nair et.al. in their on-chip test circuit. Finally, in the paper of Kinoshita and Saluja [47] a built-in-testing scheme for static adjacent pattern-sensitive has been worked out.

The paper of Westcott [45] and of Bardell and McAnney [34] are remaining papers on self-test. The first one has been criticized by Varma et al. [32] due to the silicon overhead and the fact that a complex external tester is still needed. Bardell and McAnney used their modified version of the test procedure "ATS". A slightly remarkable decision while an enlarged "MATS" pattern would be more obvious because of its simpler addressing generation, see also section 3.4.

At this point we will discuss two self-test solutions in more detail.

3.9.3 Self-Test Circuit with a PLA

Varma et al. [32] have presented an on-chip testable RAM architecture. They use a PLA to construct a finite state machine that controls the address counter and the data flow, see figure 15. The 30N algorithm of Nair et.al., see section 3.5.3, has been chosen to test the memory structure. This simple march pattern requires only an up/down address sequencer, whereas data changes can be established by means of alternating between register A and register B as data source. One flip-flop will satisfy to store the test result, because the comparator determines for each test vector whether a fault has occured or not.

During a write operation the data of the selected register is provided to the memory inputs and stored in the memory location assigned by the address counter. The data can be checked with a read action of the same location, whereas simultaneously the expected data is obtained from the same register which was originally used to write into that location. Both the data flows are routed to the comparator inputs. Only in case of a mismatch the flip-flop is set to indicate that a failure has taken place. All registers are part of a scan path, so data patterns can be shifted in, to enable a check of the test circuitry, to examine the test result and offering the opportunity to add tests on particular addresses.

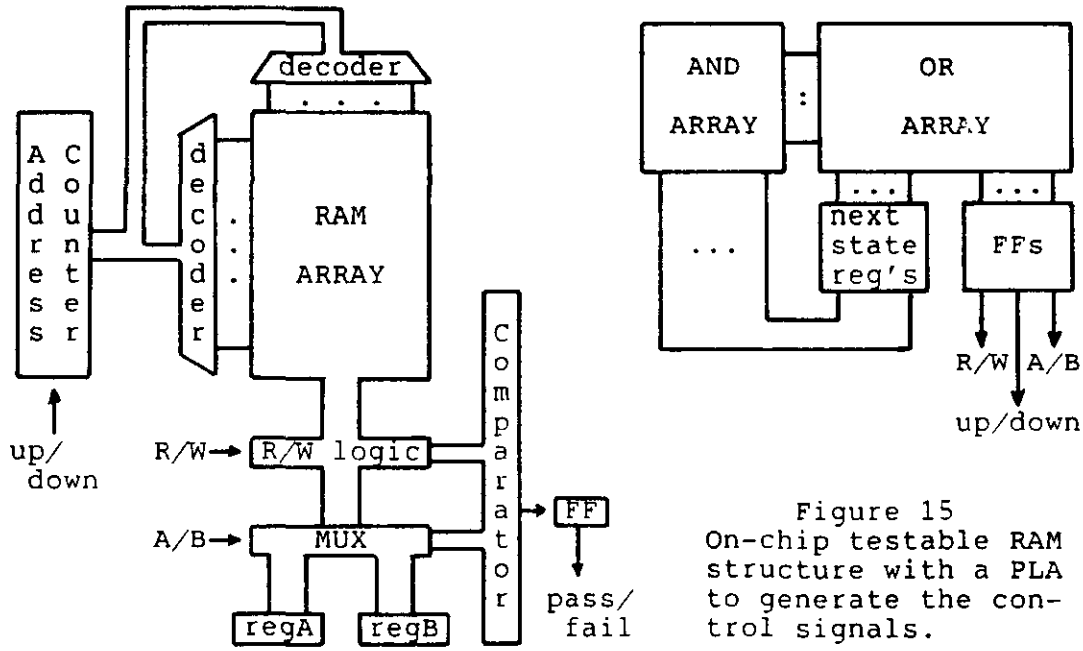


Figure 15
On-chip testable RAM structure with a PLA to generate the control signals.

The self-test scheme of Nicolaidis [48] only differs on the point of the chosen test pattern and the realisation of the control circuit. First, Marinescu's algorithm B, see section 3.5.3, was extended with two read actions in the last sequence, to obtain a symmetric algorithm facilitating its implementation. Secondly, two shift registers and some logical circuitry perform the control signals to the address counter and the test vector generator.

The self-test proposal of Sun and Wang [35] is based on similar ideas. Instead of a special address counter and data generator, they use modified address and data registers embedded in a scan chain. The MATS+ algorithm, see section 3.4, was selected, which requires only an incrementing address sequence and few data manipulations. Hence, Sun and Wang can suffice with simple adaptations of registers, which function as pattern generators during the self-test stage.

3.9.4 Self-Test Program in ROM

Kuban and Bruce made Motorola's MC6804P2 self-testable [22]. Some added ROM space stores a small program that checks all functional blocks, except for the program ROM. However, the last can also be verified in a special way and contains the customer program code, a check sum of the customer data ROM and the self-test code.

The program tests the chip in a hierarchical way. First, the CPU operations are examined including the program counter, stack pointer, addressing mechanisms and the ALU function. Secondly, the registers and interrupt logic are checked. Thirdly, the embedded DRAM is tested with a diagonal pattern, a checkerboard and complement checkerboard pattern. Fourth, the timer operation is verified. The test ends with a check of the RAM for data retention, a read of all registers and the data ROM. The finally achieved signature in a LFSR is on-chip compared with the expected value to decide whether the test failed or not.

The main difference with the previously discussed proposals concerns the employment of the program counter and some registers during the test phase. This approach saves the amount of additional circuitry for self-test purposes and gives a certain guarantee that the interconnections between the functional blocks are correct.

3.9.5 Other Developments

Testing embedded memories and, more general, the complex processor chips leads to increasing difficulties. A tendency can be visualized that even important manufacturers like Motorola tackle this problem. The most revolutionary

approach thus far has been applied in the MC68020 32-bit microprocessor [49]. During the test stage a partitioning of the architecture is forced using the bus orientation and some additional multiplexers.
This strategy opens the possibility to verify the execution unit in a purely functional way, whereas the control logic can be checked fully structurally. ROMs, PLAs and the small cache memory can be tested with a known fault coverage and at their operational speed.

However, at the moment only the manufacturer can take advantage of these measures during production testing. The user of the chip is lacking the detailed test information.

3.9.6 Considerations

Growing dimensions of embedded RAMs forces the manufacturers to make these memory structures on-chip testable. The way in which depends in the first place on the availability of other circuits like a CPU, counters, registers and secondly, on the followed test strategy.
If it concerns a very small memory, the solution used for the MC6802P2 seems preferable, whereby the verification of the RAM is included in the self-test program of the whole chip.
Larger memories c.q. complexer processor architectures demand for a furthergoing partitioning during the test phase, thus requiring a complete self-checking RAM block. This approach has also the advantage that the memory can be regarded as a standard building block with a built-in test facility. The last offers the opportunity to construct easily self-testable semi-custom chips, an aspect which comes up to the wishes of the users for more reliable components and self-checking logic.

4 MEMORY TESTER AND MEMORY TEST PROGRAMS

4.1 Introduction

This chapter discusses both the two programs that have been developed to evaluate the performance of the various data patterns encountered in chapter 3, and the automatic test equipment on which the tests have been executed. Especially the address generation, data generation combined with the verification of the data stored by the tested devices, and the timing will be described.
Next, a short survey is given of the software environment and the available analysis tools.
The chapter ends with an outline of the test programs which have been applied to a considerable number of memories. The huge information flow was analyzed with simple statistical means to trace striking memory failures and to compare the mutual detection capabilities of the various patterns. These techniques are discussed in the next chapter. The results of the first test program were employed to construct a second version.

4.2 Memory Test System

4.2.1 System Architecture

A Teradyne J389 [50] was used as memory tester. This machine contains a wide variety of hardware and software tools facilitating analysis in the engineering environment. Only the components that play an important part during the execution of the functional tests will be discussed. Figure 16 gives an idea of the hardware architecture.

The operation of the test system is controlled by software. A complete test program should contain:

a. the initial counter values of the address generator, see section 4.2.2;
b. the content to be stored in the descramble RAM, see section 2.3.4;
c. the data function provided by the data generator, see section 4.2.3;
d. the signal values and timing, performed by the wafeform generators and timing circuits of the address, data and control signals to and from the device under test, see section 4.2.4;
e. the test procedure to be stored in the pattern program RAM, see section 4.2.5.

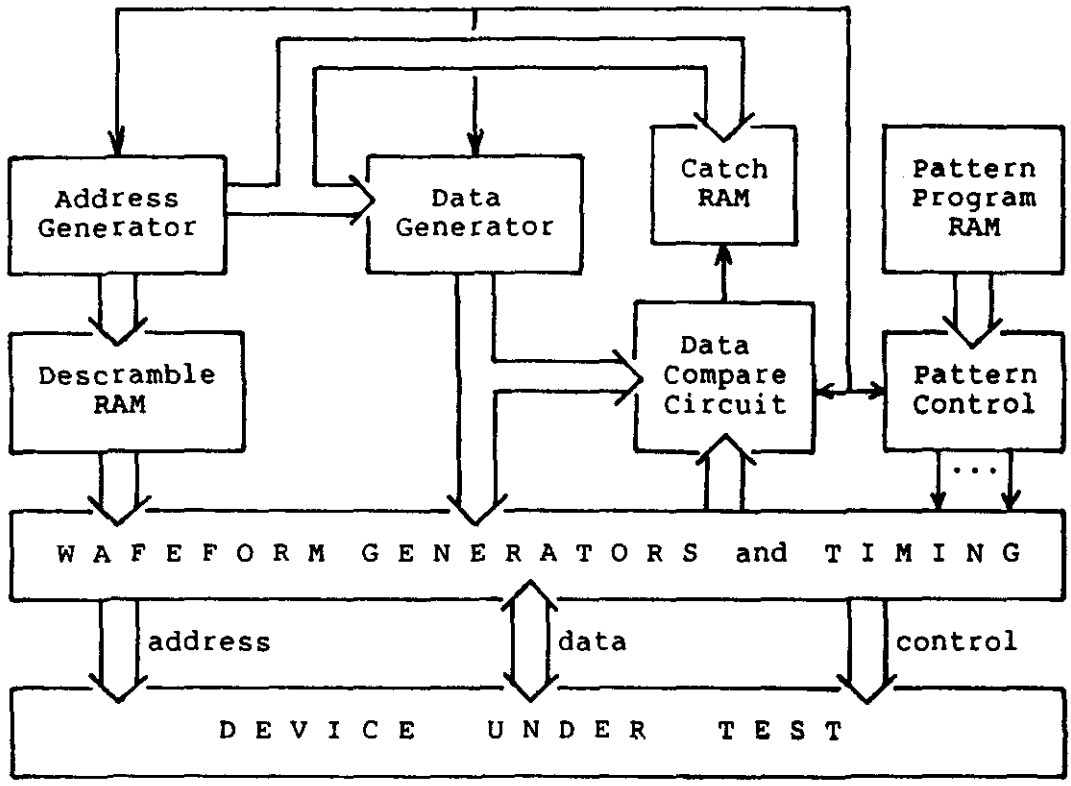


Fig.16 Internal organization scheme of the memory tester.

After the program has been loaded the system controller initializes the address counters, the signal values and timing parameters, the data function, and loads the descramble RAM and pattern program RAM. During the execution of a test the whole proces of address generation, data generation and comparison, etc., is fully in control of the pattern controller.

Overflow and underflow of the address counters as well as equalities of their values form the main software conditions either to stop a test or change the program sequence.
The data flow is straightforward during a write cycle, whereas a read operation is attended by a comparison of the expected value with the data coming from the device under test. The expected values are provided by the data generator in the same way as the data was generated during the preceding write operation. In case of a mismatch the fault memory location will be stored in the catch RAM of the tester.
The catch RAM is the internal system representation of the memory cells of the device under test. After the execution of the test has stopped each catch RAM location will indicate whether the corresponding memory cell failed during

the test or not. In turn, a counting procedure determines the final number of detected faults. The storage of the fault places in combination with their address information makes it possible to prevent double counting and to analyse the fault places afterwards.
In each cycle the signals to and from the memory are timed according to the defined waveform conditions and the current operation.

4.2.2 Address Counters

Two counters are available of which the contents can increment or decrement by only one count in a cycle. Each counter consists in fact of two identical parts, the X- and the Y-counter, to control the row decoder and the column decoder of the memory to be tested, respectively.

The address counters may be used in a number of ways to facilitate special address changes during the execution of a test pattern.
First, the X- and Y-counter part may be used as independent counters or linked to form a single counter. When the X- and Y-part are linked, only an overflow or underflow of one part causes a change of the other half.
Secondly, the configuration of the address generator may be changed in every program statement.
Thirdly, the outputs of both main counter parts can be inverted to enable jumps between memory locations with complementary address values.

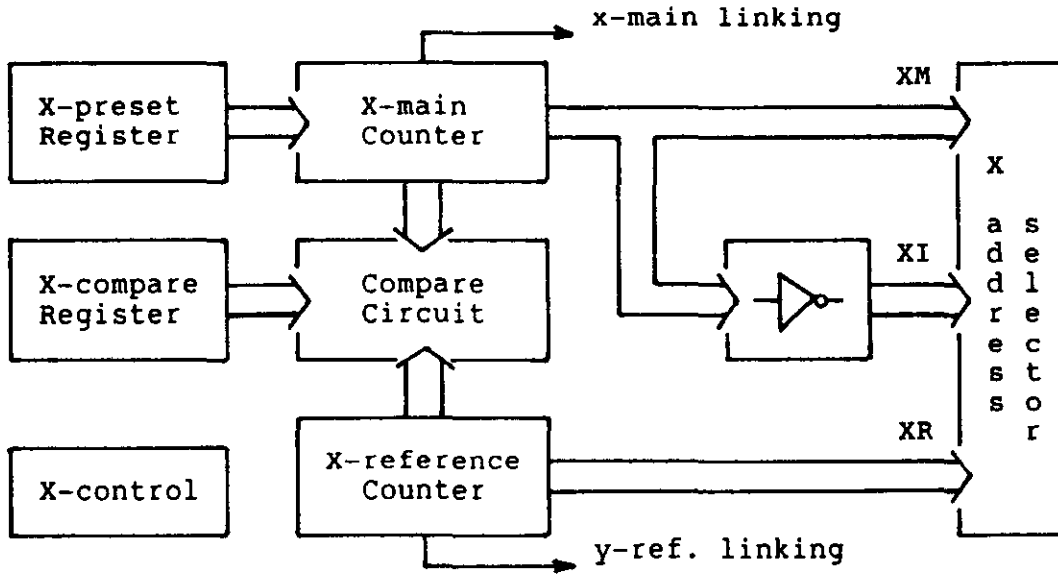


Fig.17 Counter Organization of the X-address generator part.

Above, a set of registers is present. A compare register may be used to stop the address generation if the value of a counter has reached the maximum address number. Similarly, a preset register may be used to load the main address counter with a known value on "the fly", i.e. during the execution of a test.

Figure 17 renders only the internal structure of the right half of the address generator, the X-counter parts. The other half consists of an identical counter organization for the Y-counters.

4.2.3 Data Generation

There are three sources that can provide data for the memory under test.
First, two registers may be used to store any combination of ones and zeros. Their outputs are transmitted directly to the inputs of the tested memory in case of a write operation, or to the compare circuit during a read cycle. Thus, the data is independent of the address sequence. This is called a logical data background.

Secondly, to enable very specific patterns, an arbitrary data background may be loaded in the system's data RAM. For each address the content of the corresponding data RAM location is transferred to the device under test or the compare circuit.

Thirdly, the data can be generated as function of the momentary address values. This option may be used to derive the so called topological data backgrounds. The current address values are needed to cancel internal data inversions in the memory due to address scrambling.

X_0 X_1	data = $X_1 \oplus Y_0$				data = X_1				data = $\overline{Y}_1$			
1 1	1	0	1	0	1	1	1	1	1	1	0	0
1 0	0	1	0	1	0	0	0	0	1	1	0	0
0 1	1	0	1	0	1	1	1	1	1	1	0	0
0 0	0	1	0	1	0	0	0	0	1	1	0	0
Y_0	0	1	0	1	0	1	0	1	0	1	0	1
Y_1	0	0	1	1	0	0	1	1	0	0	1	1

Fig.18 Topological data background generated as function of the current address values.

The last method is very powerful to construct efficient patterns based on the internal device structure. Figure 18 represents three examples of topological data backgrounds.

4.2.4 Signal Values and Timing

First, a set of waveform generators generates the electrical signals to the control pins of the tested device. It concerns pins like WE (write enable), OE (output enable) and CE (chip enable). For each signal the cycle time, the transition moments and the signal value have to be defined. The address and data flow towards the memory are clocked in a similar way.

Secondly, in case of a read operation the I/O pins of the device can be connected periodically with a set of load registers to simulate real circumstances. The timing moments must be initialized at which the output data is compared with the expected data generated by the system.

Figure 19 gives an example of the timing of all signals during a write action to Philips' 16K SRAM using the fast CE-latched mode [15]. Appendix C contains the three complete timing diagrams.

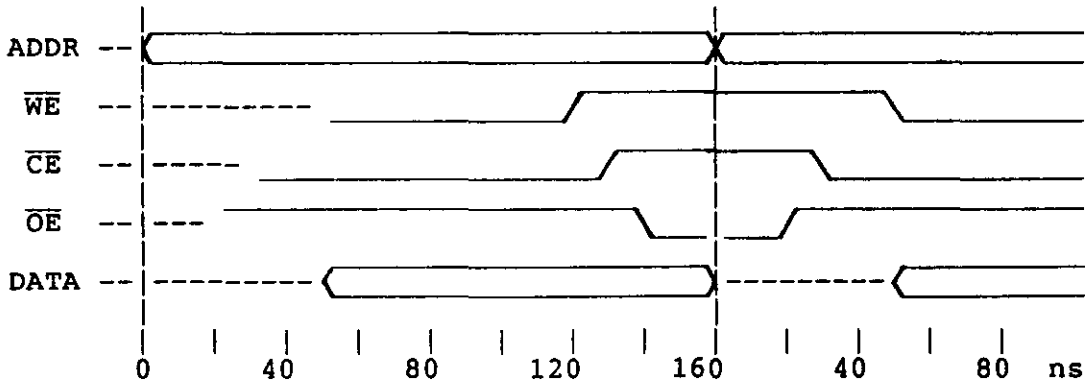


Fig.19 The timing of control lines, data and address values during a write operation with CE-latching.

4.2.5 Software Environment

Although Teradyne calls its software language PASCAL-t, the programming structure of the pattern generator has no resemblance anymore with the high-order computer language PASCAL. The language looks more like an assembler instruction set of a micro-processor. But the main critisism concerns the lack of a formal description and the ambiguity of some PASCAL-t constructions.

A typical statement to read all cells of a memory would be:

```
SOURCE XM YI  INC X DEC Y MOVE XLYM  UNTIL AM=AC BRN . MREAD
```

"SOURCE XM YI" means that the device address pins of the row decoder are connected with the normal outputs of the X-part of the main counter "XM", and the address pins of the column decoder with the inverted outputs of the Y-part of the main counter "YI".

The term "INC X" stands for an increment by one count of the X-counter parts which are enabled in the statement, whereas "DEC Y" refers to a decrement of the enabled Y-counter parts.

The enabled counter configuration is prededed by the word "MOVE". The "M" in "XLYM" means that only the X- and the Y-part of the main counter are enabled to increment or decrement. The "L" indicates that both counter parts are linked during the execution of this statement. The position of X and Y implies that the YM-counter decrements only in case of an overflow of the XM-counter.

"UNTIL AM=AC BRN ." means that the statement is repeated until the content of the entire main counter has become equal to the value stored in the compare register, "AC".

"MREAD" is a macro instruction for "CSET 0 ESET 0 FSET 0", i.e. indicating the desired timing diagram of a memory read action.

Appendix D shows the implementation of some typical patterns discussed in chapter 3. As appears from these examples, the following limitations of this programming language can be visualized:

1. As long as a simple address sequence is used, the structures of pattern programs like Y-MSCAN are simple and clear as well. However, the GALPAT and the Shifted Diagonal algorithms require tricky manipulation of the address counters to ensure well-defined stop criteria;

2. Likewise, the addressing order of the pattern GMATS-plus (Nair's MATS+ algorithm with a GALPAT like addressing method) has become very hard to understand for an outsider as result of the various program jumps and the addition of dummy statements to adjust the address counters after overflows;

3. When a number of operations are performed at the same memory location, as the pattern SUKRED does, the program becomes unnecessarily complicated due to the backwards program jumps;

4. The programming of the pattern-sensitive algorithms was hardly feasible because of the arithmetical limitations of the address counters and the restricted length of the pattern generator programs to 64 statement.

4.3 Analysis Tools of the Tester

The correctness of the pattern programs can be checked with a single step option of the pattern generator. The current value of each counter, the address source, the timing parameters and the data values can be printed. Appendix D contains two listings of the patterns Y-MSCAN and X-MSCAN, respectively. The provision is indispensable to trace bugs in the programs.

A monitor may be used to make the content of the catch RAM visible representing the memory locations which failed during a test. If the descramble RAM has been employed to generate topological addresses, it becomes possible to relate the fault places to typical device failures.

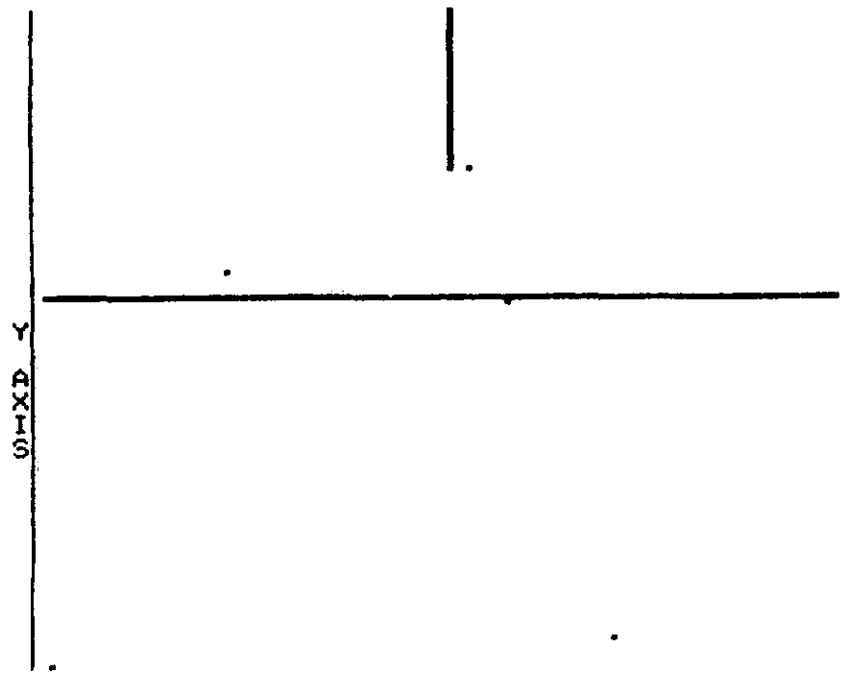


Fig.20 Bit map of a defective memory, containing single cell defects, a broken word line and a defective bit line.

Figure 20 shows an example whereby the pattern Y-MSCAN detected some single cell defects (the dots), a broken word line (the vertical line) and a defective bit line (the horizontal line) in the device under test.

Furthermore, the memory tester contains a lot of provisions which makes it a very powerful machine in the engineering environment. Timing parameters, patterns, etc., may be changed in an easy way to determine and to analyse the peculiarities of defective memories.

However, this wide variety of technical possibilities is in contrast with the bad performance of the machine's interface. On the one hand, program development has to be done largely off-line. On the other hand, operating requires a long learning period and a continuous alertness. Especially the use of the same commands within various operating windows for different actions works very frustrating.

4.4 Memory Test Programs

4.4.1 Starting Points

We saw in chapter 3 that the more efficient patterns are based on theoretical assumptions. Practical investigations were completely lacking, however. So, to get some insight into the mutual differences between the various patterns, most of them have been implemented.

The test patterns were embedded in already existing software. This saved time and the opportunity was provided to determine the shortcomings of the existing production pattern set. The last consists of the Marching 0's/1's procedure followed by the Surround Disturb pattern. This set is used for a gross functional test of the memories on the wafer in combination with DC and AC tests. In turn, after mounting the devices are tested with more extensive data patterns.

Besides, the influence of timing variations upon the detection capabilities of the patterns were analyzed. Hence, during the first stage the complete pattern set has been executed with the three timing diagrams of appendix C.

The first results were employed to construct a new test program, by which especially the effects of different addressing sequences and data backgrounds have been examined.

The figures for the investigations have been obtained through the execution of both test programs at considerable numbers of memories.

The Philips' 16K SRAM was used as test object. This static RAM has a byte-wide data bus, so there are 2K different addresses possible [15]. Unmounted devices on wafers out of different old batches have been employed to do the investigations, containing large numbers of faulty memories.

4.4.2 Memory Test Program 1

A representative set of 25 patterns was selected from the collection encountered in chapter 3. After each test the number of detected faults was printed to determine the individual detection capabilities.

Figure 21 shows the flow diagram of this test program. It starts with a continuity test after the wafer stepper has reached a subsequent memory. If all probes make connection with the bonding pats of the device, the existing production pattern set is used to determine quickly whether the device is interesting to analyze or not. Only in case of less than 16K faults the program will be continued. Both measures are taken to save test time.

After the first run the patterns GALPAT (column) and GALPAT (row) were deleted because of their extreme testing time. Although the number of memory accesses was directly proportional to 2K addresses, the execution of both patterns took about one minute each, as result of their squared complexity.

The rest of the test program consists of two main parts.
First, the pattern set is executed with the functional timing diagram, see appendix C. This gives a very relaxed clocked device so that time-dependent failures are mainly excluded. Hence, the goal of this part concerns the detection of permanent faults, i.e. the verification of the internal device structure for its logical behaviour, see section 3.2.1.
Secondly, the whole pattern set is executed twice more with the two tightened timing diagrams corresponding to the specifications, once with the CE-controlled mode and once using the address access operation mode, see appendix C. This part has been added to determine whether the tested devices are affected with AC parametric failures like write recovery and sense amplifier sensitivity, or not, see section 3.2.2.

Each time the pattern set is ended with the existing production pattern set. This measure has been taken to compare their detection properties with the properties of the other patterns. Besides, the test program starts and ends with this set, equally timed, so an adequate reference level was available to decide if the number of detected faults has changed during the execution of the tests, for

instance due to warming up.
The patterns were arranged in such a way that the lenghtened test patterns like GALPAT and GALTCOL are evenly devided among the other procedures, an attempt to exclude the effects of warming up.

The influences of other variables like signal values and data backgrounds have not been considered.

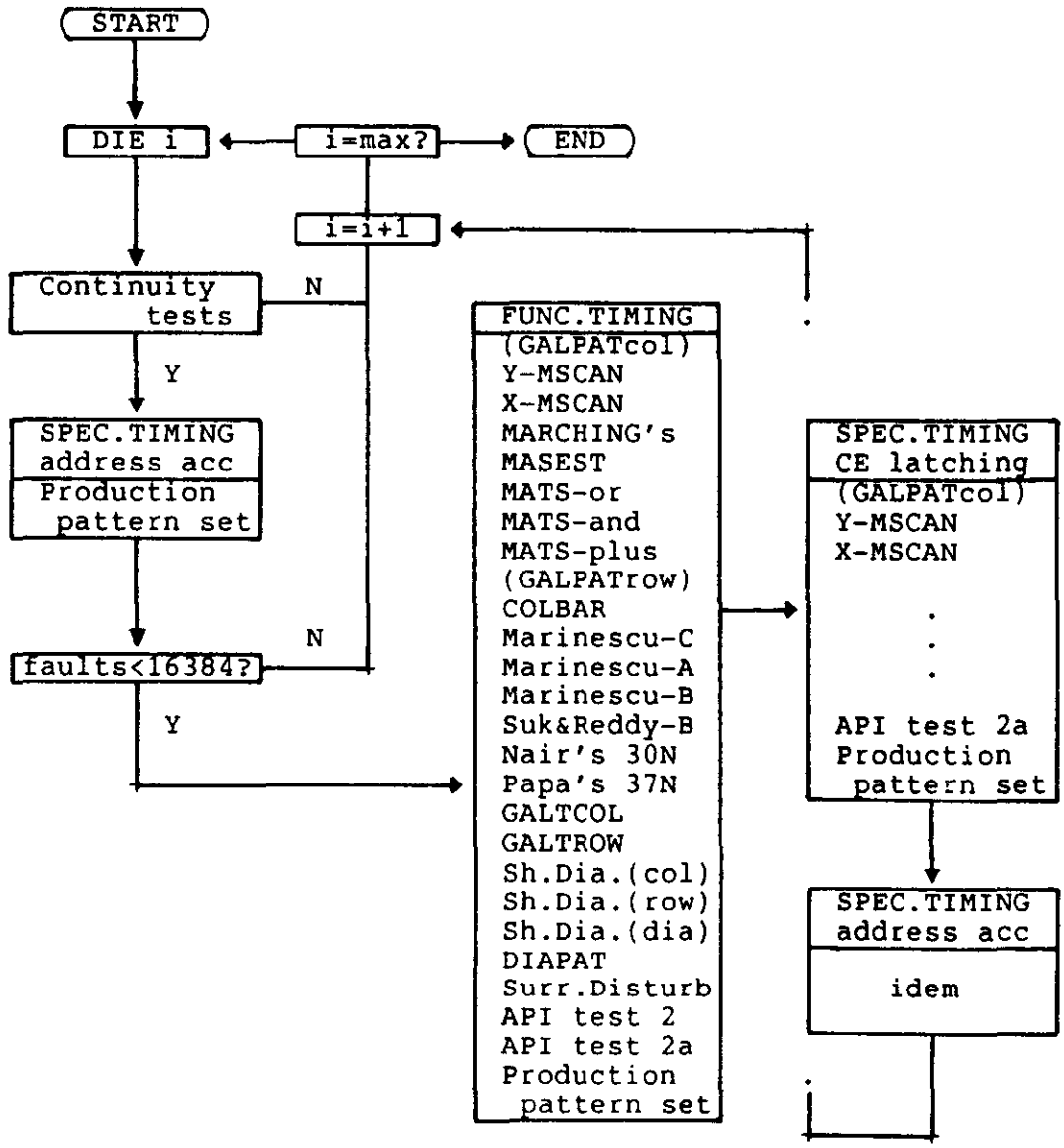


Fig.21 Program flow of test program 1 used to test Philips' 16K SRAM on the wafer. The fault number of each test is registered separately.

4.4.3 Memory Test Program 2

The structure of the second program is similar to that of test program 1, see figure 22. Some alterations were made in consequence of the results of program 1.

First, the outcomes of test program 1 were very hard to analyse due to the presence of a design error and processing shortcomings at the older wafers used for the examinations. Both effects resulted in an accumulation of time dependent fault effects, rather specific to the Philips' 16K SRAM. The main goal of this research was to make an inventory and a mutual comparison of the data patterns for the class of functional faults, independent of the typical implementation of the devices to be tested. So, the decision was made to move the priority of the investigations to a more detailed examination of the patterns using the functional timing diagram.

Secondly, the original set of 25 patterns was halved to only 12 by leaving out the patterns that showed either similar or disappointing detection properties during the execution of test program 1.

Thirdly, a dozen new patterns has been developed only differing from Nair's MATS+, Suk and Reddy's TEST B and Nair et.al.'s 30N algorithms with respect to the addressing sequence. These alterations were added to determine whether the detection properties of the pattern are sensible to the addressing order or not.

Fourth, in addition to the possible effects of address variations, also the influence of the data backgrounds has been investigated. The new set of 24 patterns was executed four times using a topological one/zero background, a topological checkerboard background, a logical one/zero background and a special one/zero background pair. The last was based on the topology of the memory cells in the memory array instead of the bit lines.

At last, most of the new patterns are executed once more with a tightened timing diagram to evaluate their speed performance. Above, some patterns were selected which showed a particular sensitiveness to the design error and processing shortcomings encountered during the first test phase. These tests are mainly added to trace the presence of these special failure origins on the wafer at which test program 2 has been executed.

The subsequent chapter discusses the results of both test programs and the analysis methods that were used to compare the performance of the various patterns.

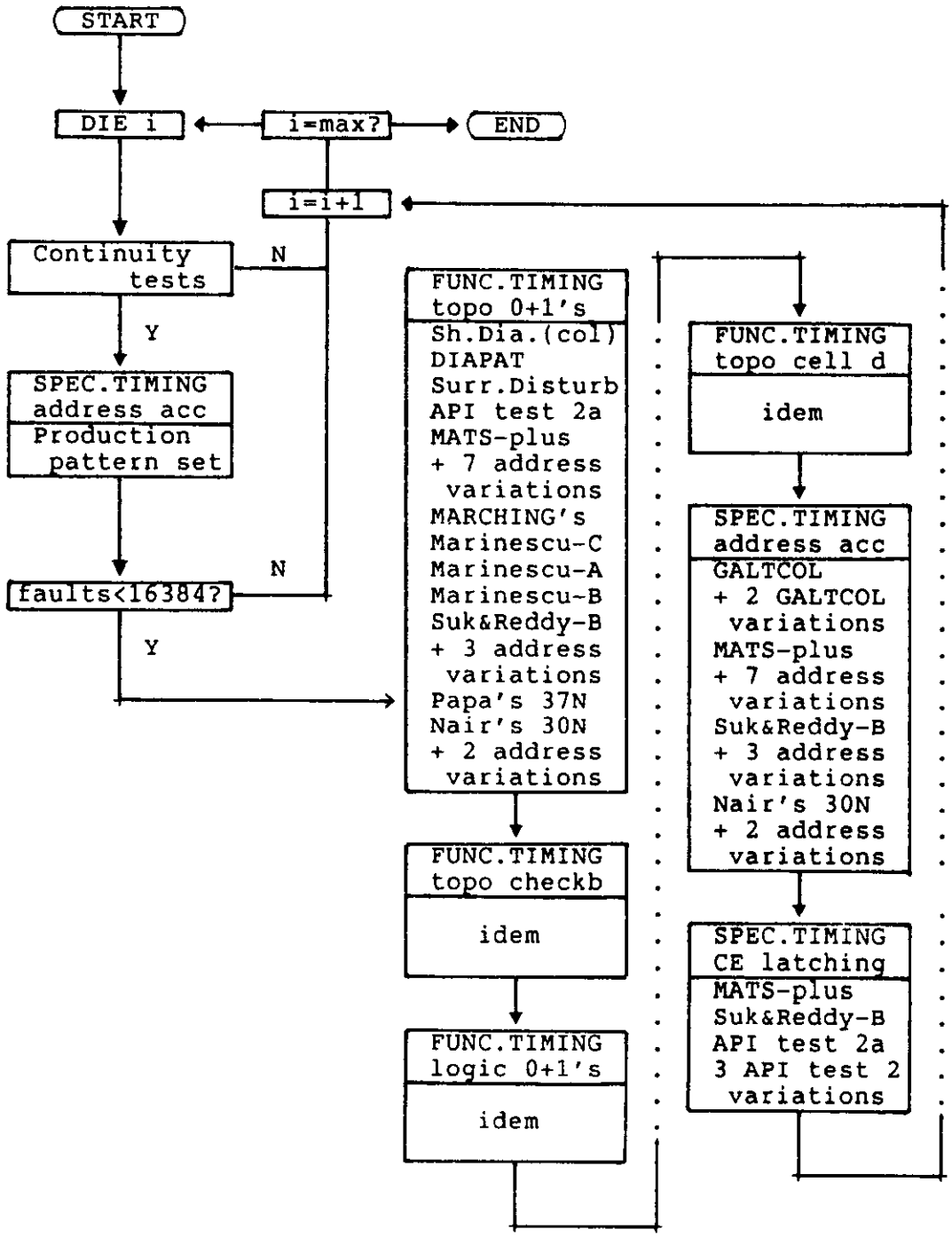


Fig.22 Program flow of test program 2. Especially influences of the data backgrounds and address variations on the detection properties are investigated.

5 DISCUSSION OF THE RESULTS

5.1 Introduction

The two test programs have been executed at several hundreds of memories. This resulted both times in enormous strings of fault numbers, caused by the fact that about one hundred numbers were generated for each completely tested device.
Therefore, simple statistical and graphical means have been applied to facilitate the analysis of the outcomes. The approach appeared very successful to make trends visible.
So, we will start this chapter with a short description of the methods that have been used.
Next, the results of both test programs are discussed separately.
The chapter ends with a summury of these results, combined with some suggestions concerning the test strategy to be used. The most suitable approach to both go-nogo tests of incoming inspection and devive characterization will be touched.

	procedure	#faults	%
1	GALPAT (column)	16384	100
2	GALPAT (row)	16384	100
3	GALTCOL	4263	25
4	GALTROW	16384	100
5	Shifted Diagonals (column)	16384	100
6	Shifted Diagonals (row)	16384	100
7	Shifted Diagonals (diagonal)	8758	53
8	DIAPAT	4172	25
9	X-MSCAN	4168	25
10	Y-MSCAN	4168	25
11	COLBAR	4168	25
12	MASEST	4171	25
13	MATS-or	13330	81
14	MATS-and	13331	81
15	MATS-plus	13330	81
16	MARCHING 0'S/1's	13331	81
17	Marinescu's alg. C	16384	100
18	Marinescu's alg. A	16384	100
19	Marinescu's alg. B	16384	100
20	Suk & Reddy's test B	16384	100
21	Papachr. & Sahgal's 37N test	16384	100
22	Nair et.al.'s 30N test	16384	100
23	Surround Disturb	4172	25
24	API test 2	4170	25
25	API test 2a	4178	25
26	Production pattern set	13339	81

Table 3 The fault number list of test program 1 for a faulty memory performed with the functional timing diagram.

5.2 Techniques to Analyse the Test Results

The first two columns of table 3 give an example of the fault number list for a selected memory. The results are obtained with memory test program 1, restricted to the part with the functional timing diagram, see section 4.4.2.
It might be clear that mutual differences between the patterns are hardly to recognize without the use of an adequate graphical technique. Four measures have been taken to simplify the analysis process.

NORMALIZATION OF THE FAULT NUMBERS

From each set the maximum fault number has been selected to scale the other values, giving the percentual distribution in the third column of table 3. The main advantage consists of the fact that from now on the new distributions are made independently of the typical device fault numbers. Hence, the normalization process opens the possibility to compare the behaviour of the various patterns across a whole wafer instead of single memories, see the third and fourth item.

PATTERN SET PERFORMANCE AT SINGLE MEMORIES

Most of these percentual distributions, like the third column of table 3, were simply plotted. Thus the distributions limited to single memories.

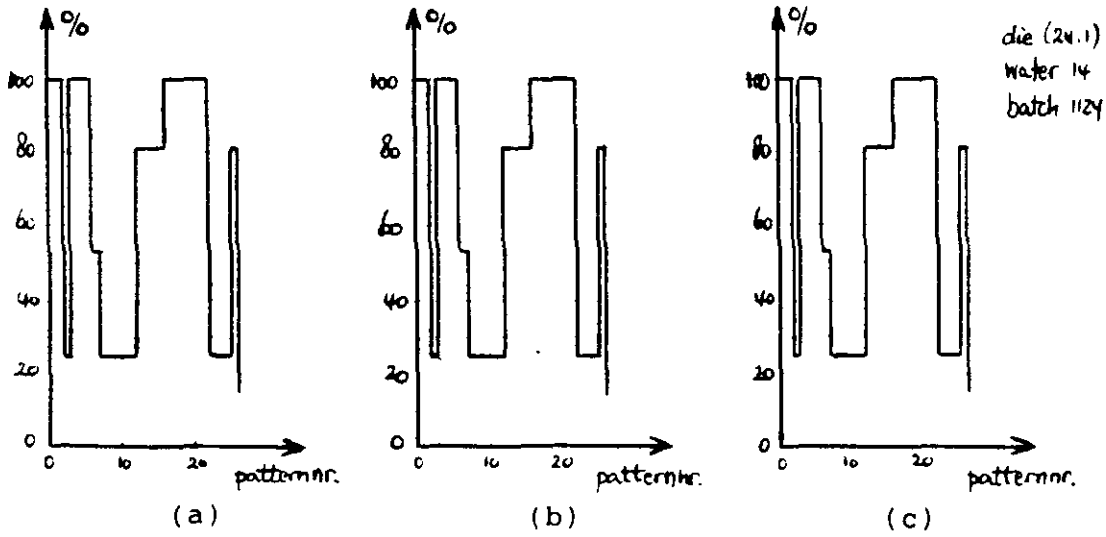
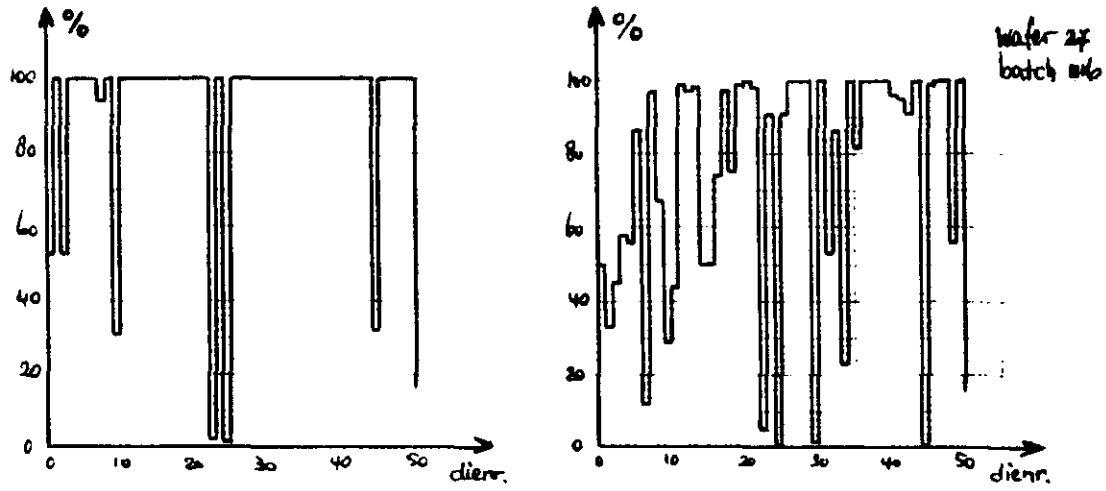


Fig.23 Mutual performance differences between the patterns at a single device for (a) the functional (%-column of table 3),(b) the CE-controlled and (c) the address access timing diagram. These three figures show the percentual distributions as function of the patterns.

The method appeared very effective to select only those memories for which the patterns showed a remarkable and consistent difference in performance, i.e. independent of the timing conditions, see figure 23.

PATTERN PERFORMANCE ACROSS THE WAFER

Similar plots were made for each pattern as function of the devices on a wafer, see figure 24. Hence, the possibility is offered to compare the detection capabilities of the various patterns across a whole wafer instead of single memories. Especially shortcomings at particular memories are easy to recognize as dips of these distribution functions.



(a) (b)

Fig.24 Detection properties of (a) the GALTCOL and (b) the Y-MSCAN pattern accross one wafer as function of the memories tested. Both patterns were executed with the same timing diagram.

AVERAGE PATTERN SET PERFORMANCE ACROSS THE WAFER

The average performance of the pattern set was determined through a second calculation. For each individual pattern the normalized fault numbers, related to the same wafer and timing conditions, were added and devided by the number of memories involved. This gives the average performance distributions of figure 25. The plots turned out extremely useful to demonstrate design errors, processing problems at wafers, differences between fault models and the influence of a data background on the detection properties of the patterns.

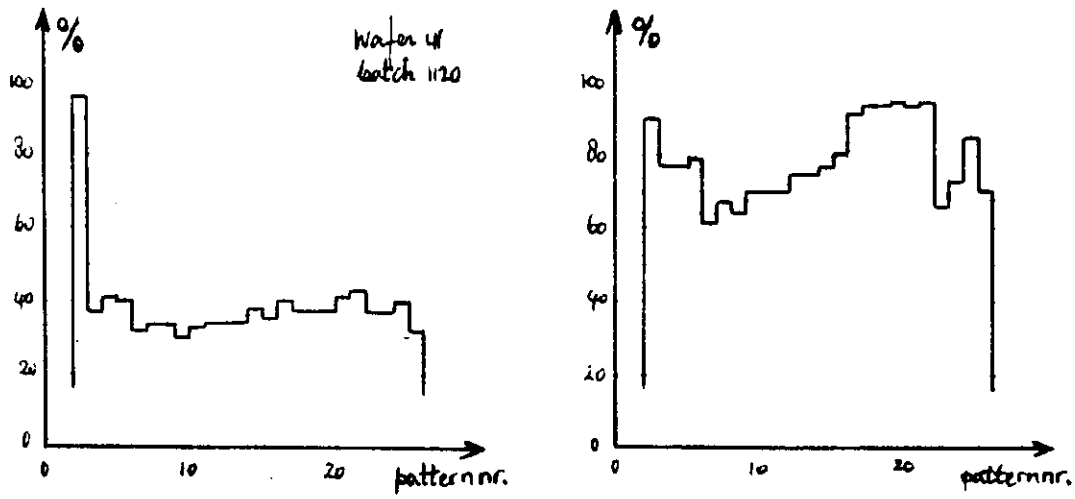


Fig.25 Average performance of the pattern set at the same wafer with the (a) functional and (b) address access timing diagram.

Although these simple operations are easy to program, some practical problems with the communication link between the memory tester and the VAX computer environment hindered an automated processing of the data, unfortunately.
Consequently, all calculations and drawings had to be done by hand, that is why more laborious manipulations were left out of consideration.
Above, only those memories have been considered for which the patterns showed a differing detection capability. This limited the number of calculations to about a third.

5.3 Results of Memory Test Program 1

Three old wafers of Philips' 16K SRAM have been used to investigate the mutual detection properties of the patterns with test program 1, see also section 4.4.2.

In first instance a design error and some processing problems clouded the outcomes, especially in case the timing was tightened. Detailed examinations have been carried out to ascertain the origins of these irregularities. It turned out that the confusing results were caused by time-dependent faults.

In second instance the attention was fixed upon the more reliable outcomes of the functional program part, i.e. the permanent memory failures.

5.3.1 Discussion of the Time-Dependent Faults

By a time-dependent fault is meant a fault which manifest itself due to either a change of the operation mode or an alteration of the cycle time. Consequently, these faults are very specific to the device type as well as the wafers that have been examined.
A short description will be given of both three striking failure types encountered at these three old wafers, and a programming mistake, namely:

1. A design error resulted in an increased detection capability of the patterns with a bit line following addressing sequence. The phenomenon became visible when the address access timing diagram was used;

2. Processing shortcomings led to a remarkable detection sensitivity of one pattern in particular. The pattern contained a very specific data and addressing order. The problem appeared with the shortened CE-controlled timing diagram;

3. The patterns with a word line following addressing order showed an improved performance when the functional timing diagram was used;

4. A programming mistake caused that some patterns detected significantly more faults;

1. A design error caused the patterns with a bit line following addressing sequence to detect significant more faults:

ad.1 The design error makes that the memory cells around the X-decoder are too fast. These cells become accessible before the complete address is encoded, due to an internal device timing irregularity. This happens only in case the memory is controlled according to the address access operation mode (appendix C), independent of the cycle time.
In particular data patterns whereby the addressing sequence follows the bit lines, see the description of the X-MSCAN pattern in appendix D, displayed a significant better detection capability. Figure 26 gives an idea of the resulting overall distributions as it was encountered at two wafers. The more detailed plots 1 and 2 can be found in appendix E.

The fault effect results in an excessive detection superiority of the GALTCOL pattern. A striking difference, especially when one compares the performance of this pattern with that of the GALTROW pattern. The last pattern contains a jumping addressing

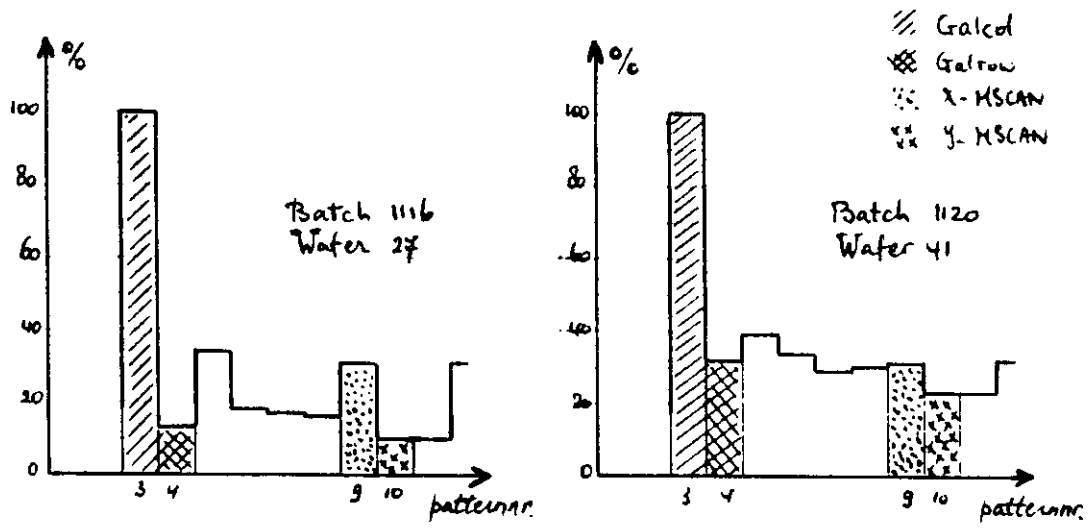


Fig.26 The effect of a design error results in a superiority of the GALTCOL algorithm when the address access mode was used. The fault effect is more dominating in the left picture, resulting in lower performance figures of the other patterns.

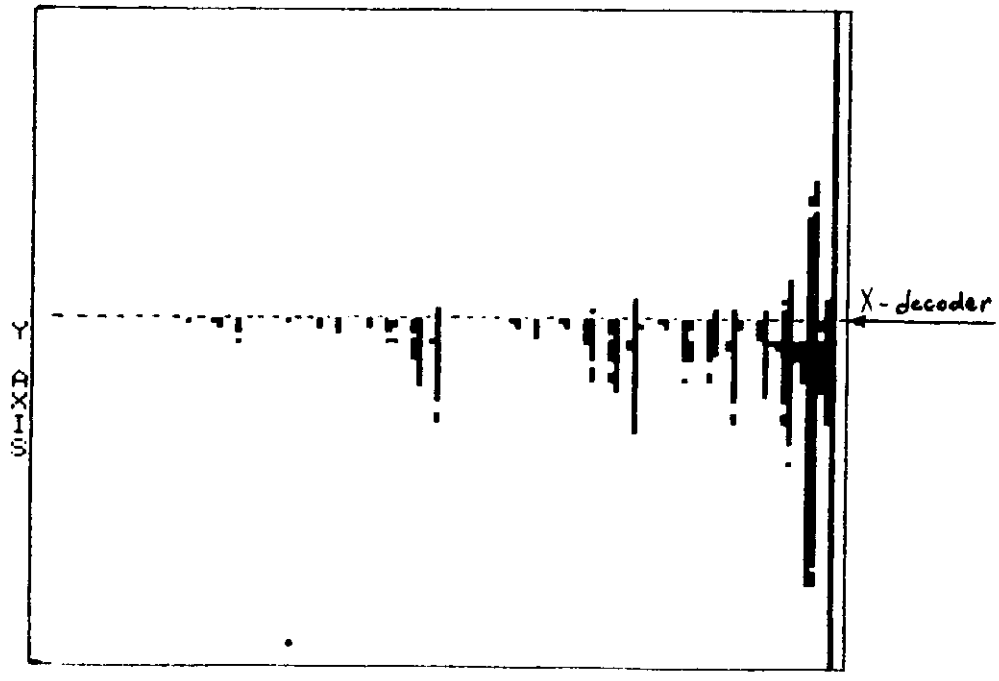


Fig.27 The address access problem results in the malfunction of the fastest cells located around the X-decoder.

sequence between memory cells sharing the same word line, whereas the addressing jumps of the GALTCOL algorithm are restricted to cells sharing the same bit line.
But even simple marching patterns like X-MSCAN and Y-MSCAN show a remarkable difference in detection, only explicable on the ground of the bit versus word line following addressing sequence.
Figure 27 gives a representative picture of this characteristic fault effect, which was obtained with the GALTCOL pattern.

Although the problem is of no practical importance anymore, the distributions of figure 26 make clear that the execution of a large pattern set in the engineering environment may help to trace design errors.
Patterns showing an extreme detection sensitivity, might shorten the analysis process. In turn, with the aid of the distribution functions of single memories, like figure 23, those devices can be selected most affected by the fault effect. The last are most interesting for further research.

2. Processing shortcomings led to a remarkable detection properties of one particular pattern:

ad.2 This phenomenon appeared when the pattern set was executed with the CE-controlled timing diagram, see appendix C. The API test 2a for pattern-sensitive faults displayed substancial larger faults numbers.

The explanation of the fault origin was in this case less obvious. The very strange pictures, see figure 28, couldn't be easily related to the regular device structure. More detailed examinations of some memories learned, that processing shortcomings lead sometimes to a sharp rise of the substrate voltage.
The shortened cycle time gives fast switching of the I/O circuits by what many hot electrons are injected in the substrate. In those cases the back-bias generators were probably no longer able to prevent the voltage rise. This malfunction resulted in the random destruction of the memory cells contents. Especially the cells located near to the device peripheral circuits were affected by the fault effect. The pattern in question contains a deviating addressing and data sequence that caused the sympton to manifest.
Plot 3 in appendix E gives the corresponding average performance distribution of the most affected wafer. One can observe the superior performance of the API test 2a pattern and an improved detection capability of Suk & Reddy's test B as well.

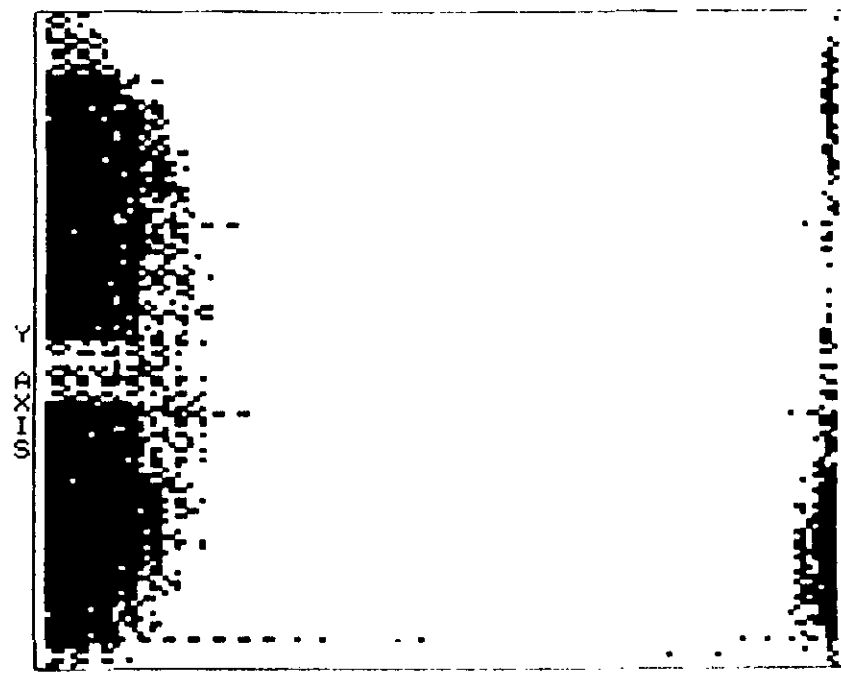


Fig.28 The rising substrate voltage makes that the contents of the cells near to the I/O circuits are destroyed.

Hence, this specific fault effect demonstrates again the strength of a large pattern set to trace time-dependent faults.

3. The patterns with a word line following addressing order displayed an improved performance using the functional timing diagram:

ad.3 The patterns displayed remarkable differences in performance at the nearly good devices. On the one hand procedures with a word line following addressing sequence, see the description of the Y-MSCAN pattern in appendix D, detected at wafer 14 of batch 1124 larger amounts of faults than their word line following counterparts did, compare the results of pattern GALPAT (row) vs. GALPAT (column), GALTROW vs. GALTCOL, and Y-MSCAN vs. X-MSCAN in plot 4 in appendix E. On the other hand the percentual distribution of wafer 27 of batch 1116 shows a pretty smooth behaviour with only small ups and downs, see plot 5 in appendix E.
Hence, the superior detection properties of the patterns with a bit line following addressing order for the address access design error have changed in favour of the patterns with a word line following addressing

sequence, compare plot both 1 vs. 4 and plot 2 vs. 5 in appendix E.

The first wafer contained a lot of memories having a random distribution of single bit errors. Even stranger, the fault numbers often reduced at these devices after the cycle time had been shortened. That points certainly to data retention problems, i.e. individual cell contents have become volatile due to processing imperfections. This elucidates the reducing fault numbers due to a shortened timing.

Also an explanation can be found based on the internal device functions. The selection of a word line makes that all cells in that row becoming accessible. In turn, each access also results in a sort of refresh operation of the semi-static RAM cell. So, small time intervals between the activations of each word line gives the volatile cell contents a better change to survive.
This makes that the elapsed time between two activations of the same word line will be much shorter for a bit line following addressing sequence than with a word line following addressing sequence. Hence, if a device suffers from data retention, the GALPAT (row) and the Y-MSCAN patterns will detect much more faults than the GALPAT (column) and the X-MSCAN pattern, respectively.

We still have to elucidate the striking performance differences between the other patterns, see plot 4 in appendix E. However, these differences can also be related to the data retention defects.
Above, the fault effect was less predominent at the other wafer, which explains the pretty smooth character of the corresponding average pattern set performance, see plot 5 in appendix E.

4. Discussion of a striking programming mistake:

ad.4 This happened during the first run of test program 1. A few patterns detected significant more faults if the CE-controlled operation modes were used.
The programming mistake altered the CE-controlled read and write operations into memory actions with address access when they were preceded by a dummy statement. This resulted in a mixture of detected faults, namely faults due to the address access problem occured even when the CE-controlled mode was used.
Only the limited set of patterns containing the dummy statements viewed the increased fault numbers, by which the mistake was located. After the mistake was corrected, the patterns showed a normal behaviour.

5.3.2 Discussion of the Permanent Faults

The outcomes of the functional program part were better explicable with respect to the underlying fault modelling. Appendix E contains the average pattern set distributions of two wafers, plot 4 and 5. The distributions have been been splitted up into a plot of the nearly good memories and a plot of the remaining faulty devices, the categories less than 300 faults and more than 300 faults respectively. By that both collections refer to about an equal number of memories.

By means of the percentual distributions related to single memories, see figure 23, a selection has been made of those memories at which remarkable detection differences between the patterns could be observed. Especially devices whereby these three performance plots show a strong resemblance caught the attention, i.e. indepent of the timing diagram employed.
This limited set of devices has been examined in more detail, searching for explanations of these consistent differences.

The experiments led to the following three observations:

A. The average performance of the pattern set can not be related to the fault models as long as the memories contain only a limited number of faults;

B. The more serious injured memories show the gains of a fault model, especially in case of decoder failures and coupling effects;

C. the existing production test pattern set may be simply improved in the sense of fault coverage and efficiency.

ad.A At this point only the devices mainly affected with single bit and/or single line errors will be discussed, thus the category with less than 300 detected faults.
Then, even simple algorithms like MSCAN display a moderate score, see plot 4 and 5 in appendix E. This result is not surprisingly, because the correctness of the decoder and the read/write logic may be assumed for this category of devices. In fact, faults in the periphery manifest themselves into strings of completely failing rows and/or columns.
Hence, the detected faults refer to stuck-ats of single memory cells and couplings between neighbouring cells. Since each pattern contains the necessary operations to detect stuck-ats, the first failure type, i.e. the stuck-ats, will be covered by all. This explains the moderateness of the average pattern set performances, see plot 4 and 5.

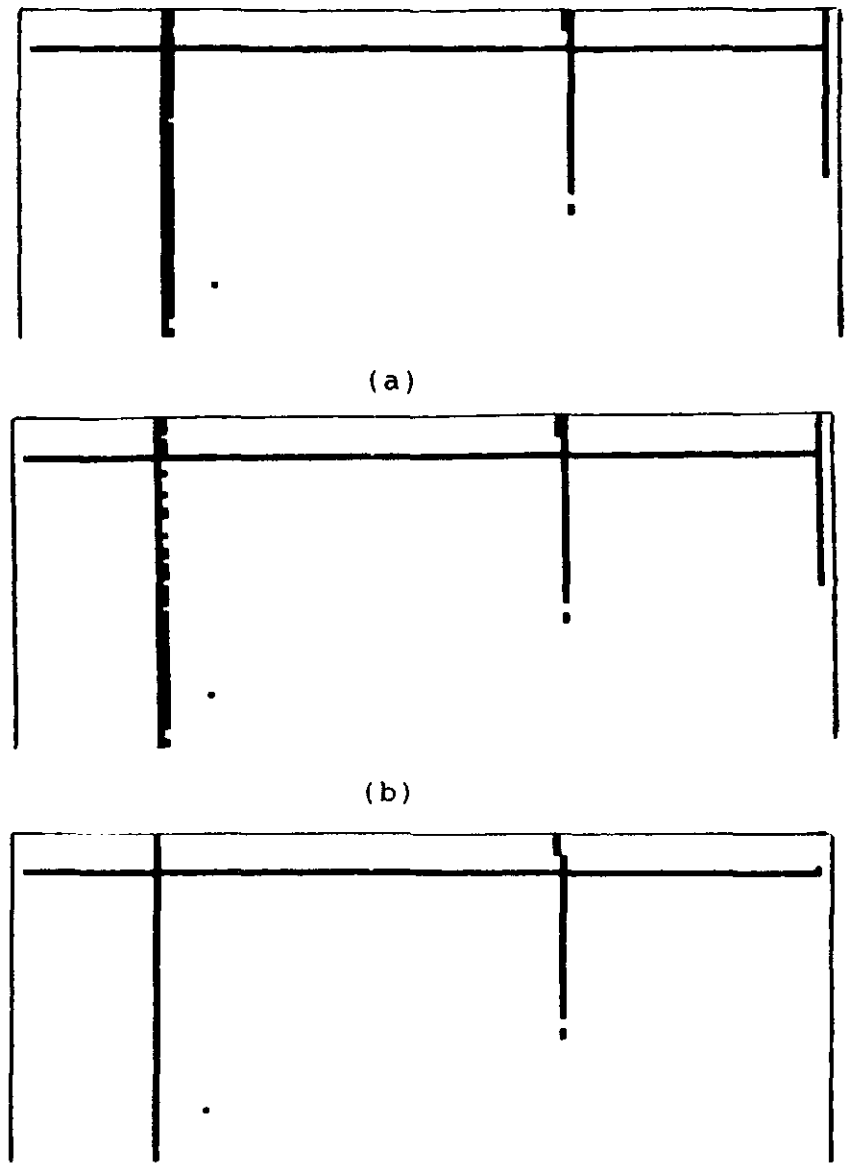


Fig.29 The coupling influences between cells sharing the two neighbouring word lines in the upper half of the memory array have been detected best by (a) Suk&Reddy's test B, followed by the (b) MATS+ procedure, whereas the (c) Y-MSCAN procedure missed both these couplings and the broken word line at the right.

On the contrary, coupling faults can only be detected by the more sophisticated test algorithms. Although the vast majority of the faults consisted of single

stuck-ats and single lines, a few memories have been found containing faults that support the coupling assumption.
An example is given in figure 29. The 2-coupling pattern Suk & Reddy's test B detected more coupling faults than the MATS+ pattern did, whereas the Y-MSCAN procedure was completely failing.

However, a strict relation between the pattern performances and the underlying fault modelling could not be discovered. Although the MATS patterns showed an improved detection capability over the MSCAN procedures, no further improvement could be observed using the more extensive tests for 2-coupling faults. outcomes.
Even the pattern-sensitive algorithms did not display an increased performance, notwithstanding their specific suitability to detect most kinds of coupling influences between neighbouring cells.

B. The more serious injuried memories show the gains of the fault modelling:

ad.B The pattern performance at the serious injured devices shows a radically changed picture, see the plots 4 and 5 in appendix E with more than 300 faults.

First, the ad hoc procedures DIAPAT, X-MSCAN, Y-MSCAN, COLBAR, MASEST and Surround Disturb make a considerable decline in fault coverage. Even the time consuming GALPATs, GALTCOL, GALTROW and the Shifted Diagonals do not excel and display a highly addressing dependence.

Secondly, both the pattern-sensitive procedure API test 2 and the extended version API test 2a are disappointing. The result makes clear that these memory cells aren't sensible to neighbouring parasitic coupling effects, which comes up to the expectations for a static memory.

Thirdly, a drop in performance can be visualized for the MATS and the Marching 0's/1's procedures, but the mutual differences are negligible. The last result seems remarkable, since the Marching 0's/1's algorithm takes 14N memory operations against 4N or 5N for the MATS procedures. However, their fault model is similar, which explains the resemblance.
On the contrary, the patterns 17 up to 23 display excellent scores. The improvements are related to the properties of the underlying fault model, i.e. the results affirm Thatte's assumption that decoder and read/write failures can be translated to faults in the memory array.

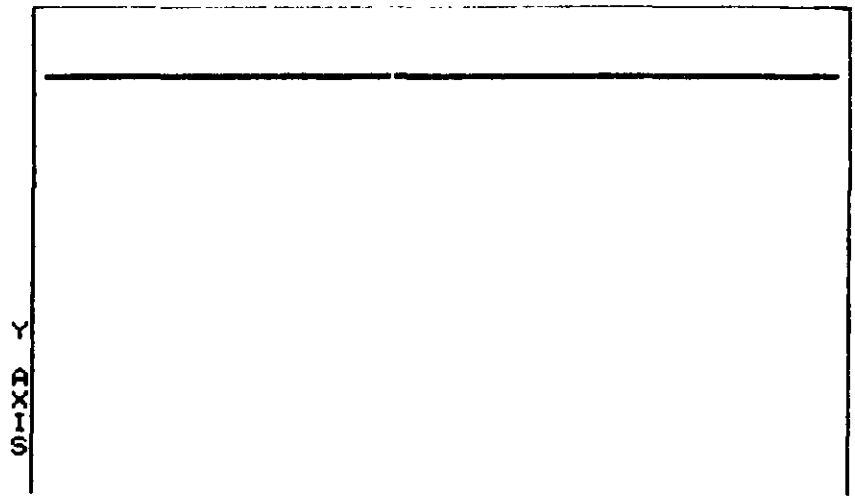
Fourth, the fault detection properties of the patterns 17 up to 23 bear a remarkable resemblance. Papachristou & Sahgal's 37N test and Nair et al.'s 30N test do not display a significant improvement in comparison with the other four patterns.
Thus, the simultaneous presence of several types of failures does not influence their fault coverage. Hence, we may suffice with the less time consuming algorithms proposed by Marinescu and Suk & Reddy, instead of the comprehensive 30N or 37N tests. See also section 3.5.3.

The figures 30, 31 and 32 show the fault pictures of three different memories. They are discussed in short.

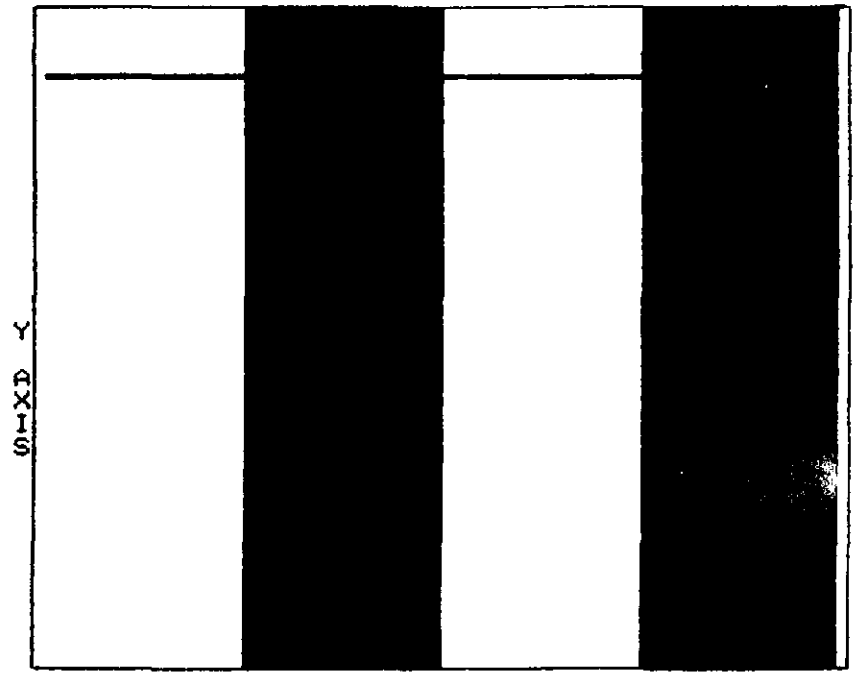
First, all patterns detected a faulty bit line at the memory corresponding to figure 30. The white planes in (a) and (b) assure that the Y-decoder and the read/write logic are fault free. The regular located black rectangles in (b) point to a malfunction of the X-address circuitry.
The most probable origin is a failure of the most significant X-address, either caused by a malfunction of the address buffer or due to a faulty address line.

Secondly, both fault pictures of figure 31 show a X-address problem in the lower array half, the vertical bars. Above, the Y-decoder of the lower half seems broken, because of the regular location of the small horizontal bars, see (a). Also a more serious failure of the Y-address logic has been detected, completely by all 2-coupling tests and partly by the MATS and the Marching 0's/1's procedures, see (b). The read/write logic functioned correctly, indicated by the white planes in (a) and (b).
However, the location of the black planes of (b) can not be explained since both Y-decoders are mirrored around the X-decoder. A possible origin would be the mistakenly switched-off address descrambler.

Thirdly, the memory corresponding to figure 32 was probably affected by failures in the I/O logic or Y-address decoding function. This explains the regularly repeated horizontal lines in both array lines, see (a). The MATS, the Marching 0's/1's and the 2-coupling patterns point to more serious faults of the Y-address decoding function, since at least three fourths of all cells are fault.
Again, the location of the black planes in (b) can not be elucidated due to the mirrored location of the Y-decoders.

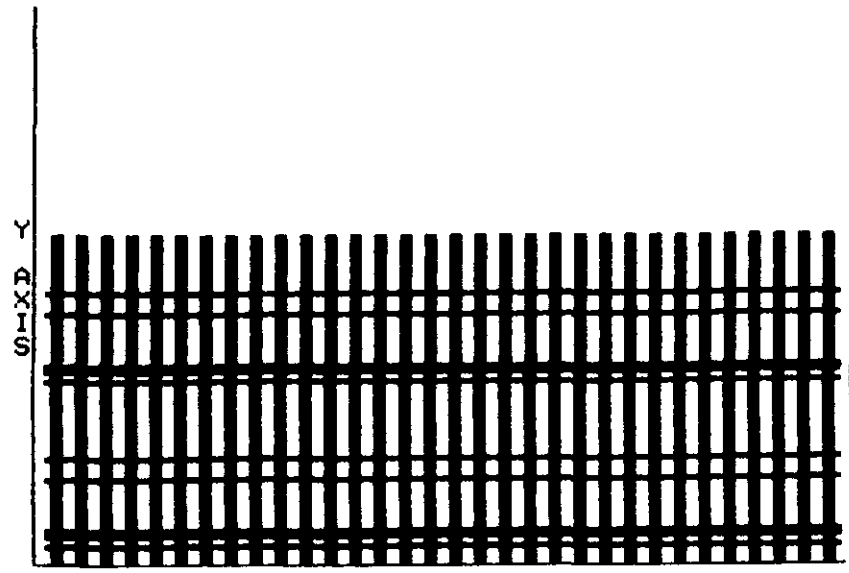


(a) The patterns GALTROW, Shifted Diagonals, DIAPAT, MSCANs, COLBAR, MASEST, Surround Disturb and API test 2 detected 128 faults, corresponding to a faulty bit line.

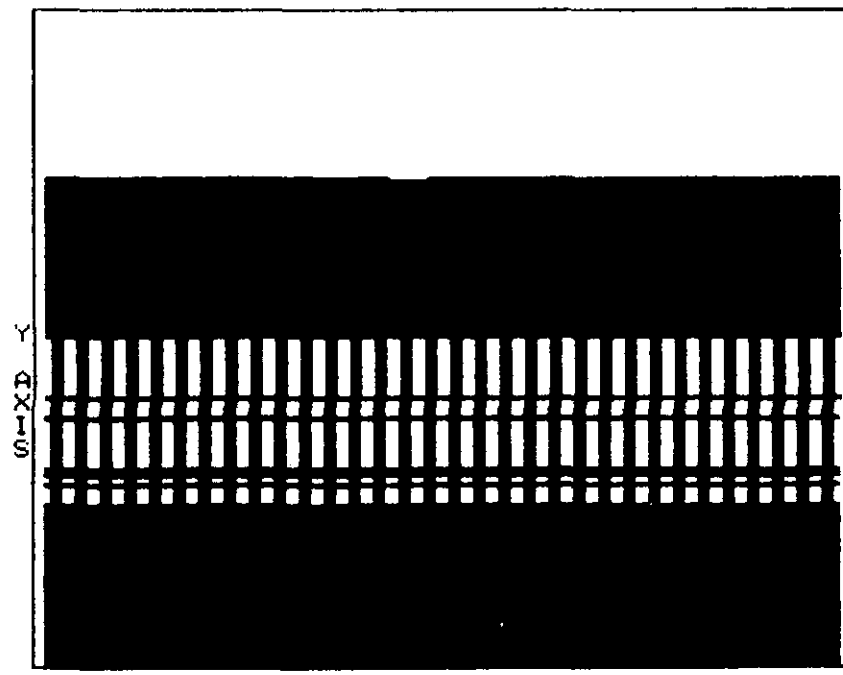


(b) The MATS and Marching 0's/1's patterns detected next to the faulty bit line a malfunction of the X-address decoding function, resulting in 8256 faults.

Fig.30 Pictures of a memory with a failing X-decoder and one faulty bit line. The pattern GALTCOL and the coupling algorithms proposed by Marinescu, Suk and Reddy, Nair et.al., and Papachristou and Sahgal showed the fully failing X-address decoding, they detected 16K faults.

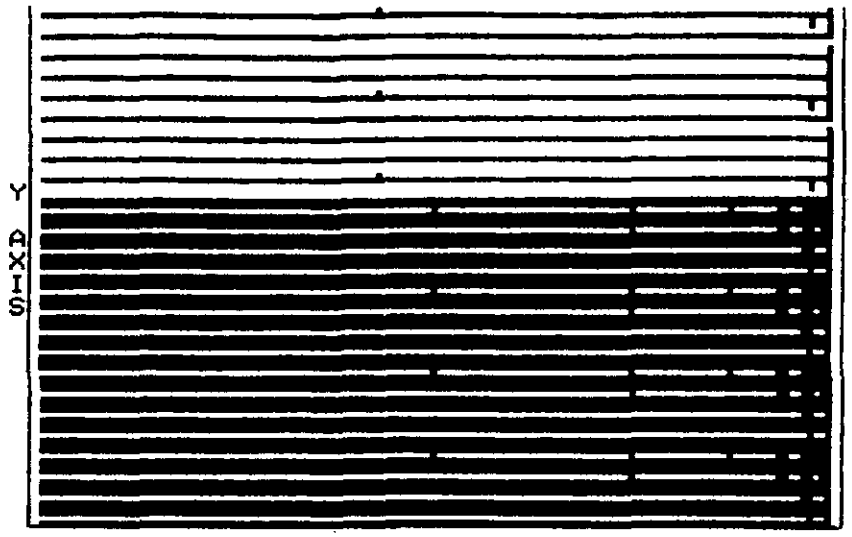


(a) The patterns GALTCOL, DIAPAT, MSCANs, COLBAR, MASEST and the pattern-sensitive tests detected only the addressing problems in the lower array half, giving 4736 faults.

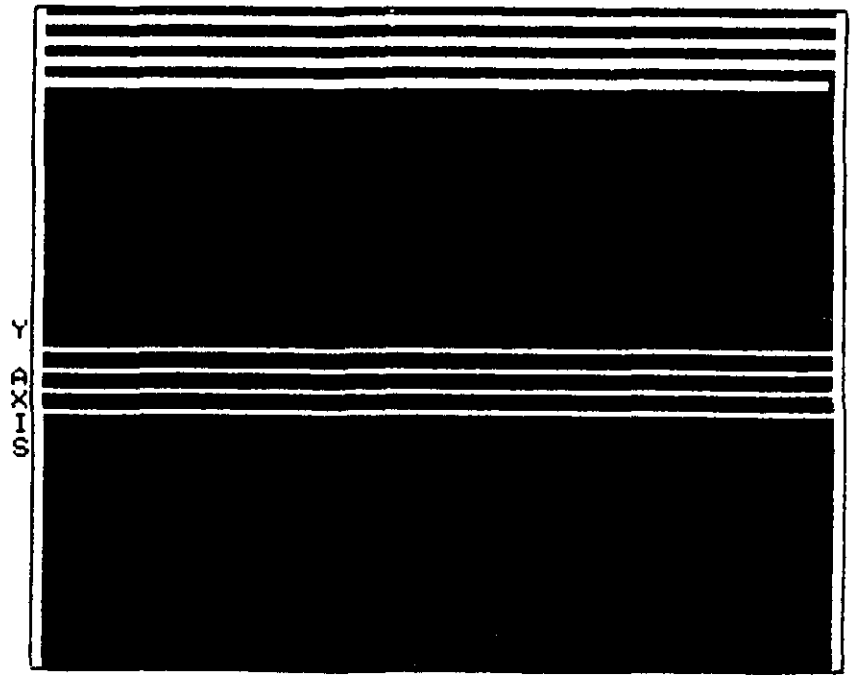


(b) The MATS and Marching 0's/1's procedures showed problems of the Y-decoding function, and detected 10560 faults.

Fig.31 Fault pictures of a memory with a completely failing Y-decoding function, because the GALTROW pattern and the 2-coupling algorithms detected 16K faults.



(a) The patterns GALTCOL, DIAPAT, MSCANs, COLBAR, MASEST and the pattern-sensitive tests only display some addressing problems in the Y-decoder or failures in the read/write logic. They detected about 8200 faults.



(b) The MATS and Marching 0's/1's procedures also sentisized a more serious Y-address fault, leading to 14340 faults.

Fig.32 Pictures of a faulty memory at which the GALTROW and the other algorithms detected 16K faults.

C. Some alterations may improve the production pattern set:

ad.C A description of the existing production pattern set and its application can be found in section 4.4.1.

As expected, the detection properties of this pattern set disappointed somewhat at the more injured memories, but in accordance with the previous observations. The performance looks most like that of the Marching 0's/1's pattern. The small differences are probably the result of the more effective data backgrounds with which the individual Marching outcomes have been obtained, see the next section.

Opposite, the Surround Disturb part seems completely superfluous, despite of its 24N memory accesses. However, the result is not surprisingly, because the pattern contains a rather ad-hoc read/write sequence to detect pattern-sensitive faults.
Although the four surrounding nibbles of a test nibble are written with opposite data and rewritten with normal data, each write operation is not succeeded by a read action. That makes that most of the neighbouring coupling faults will not be detected by the Surround Disturb pattern as the previous discussed pattern-sensitive tests do.

Hence, the existing pattern set appears to be improper and asks for some improvements. These conclusions are easily be explained due to the fact that these patterns were originally developed to test dynamic instead of static memories.

5.4 Results of Memory Test Program 2

A description of the program and the research goals can be found in section 4.4.3.
This program has been executed on a shade better wafer, without the predominating address access design error and the substrate voltage problems, which hampered the outcomes of test program 1 to some degree.
The wafer that was selected, contained a large amount of nearly good devices, making it possible to investigate the influences of address and data variations in more detail.
The memories have been devided into a category with no more than or equal to 10 detected faults and a category with more than 256 faults. By this, both categories refer to comparable numbers of memories.
The clear results led to a deeper understanding of the diverge failure mechanisms, which in turn has helped to explain the outcomes of test program 1.

5.4.1 Discussion of the Time-Dependent Faults

A definition of the time-dependent faults was given in section 5.3.1.

The patterns that showed an extreme sensitivity at the previous wafers for the address access error and the rising substrate voltage, have been selected again to determine whether this wafer was affected with the same problems or not.
First, the GALTCOL pattern has been used to investigate the address access problem. It turned out that some devices still showed some address access faults, i.e. the nature of the problem has become less serious and the effect was only restricted to a limited number of devices. Simple attemps to half the complexity of the GALTCOL pattern were not successful.
Seconly, the rising substrate voltage problem has not been found.

Hence, two of the three striking time-dependent fault effects did not influence the experiments of test program 2 anymore. However, the data retention problems were still present.

Eight addressing structures have been programmed to examine this aspect, near to possible influences of the addressing order on the detection of permanent faults. Figure 2 and 3 in appendix F give an idea of the successive assignments by which the memory cells are accessed. They range from a ordinary bit or word line following order to a GALPAT-like jumping addressing sequence. The MATS-plus algorithm has been selected to compare the eight methods, because of its simplicity.
Suk & Reddy's test B and Nair et al.'s 30N algorithm are also executed with some address variations to investigate the consistency of the results obtained with the MATS-plus pattern.

The outcomes of the test program led to the following conclusions:

> The addressing order has a large impact on the performance of the patterns, when they are executed with the functional timing diagram and as long as the number of faults is limited.
> The pattern performance is similar when the memories are executed with a tightened timing diagram.

If we compare the performance of the eight MATS-plus variations, differences up to 25% can be observed, see plot 1 in appendix F of the category no more than or equal to ten faults.

More curious seems the fact that the largest discrepancy appeared between the patterns with a bit and a word line following addressing sequence. Nevertheless, the origin proves to be the same data retention problems as we already encountered in the previous section. Even the slightly lower score of the B-version with regard to the D-version, can be explained by the smaller number of address transitions between cells sharing the same word line.

The patterns based on the coupling fault model show similar effects.
However, in case of Suk & Reddy's test B the address variations have a smaller influence on the performance than the MATS-plus procedure and Nair et al.'s 30N test displayed. The effect is probably caused by the fact that Suk & Reddy's test B has a better score with a bit line following addressing order. In turn, this improvement comes on the account of the longer operation sequence at each memory location, which results in a better detection of the data retention problems. The same effect can be recognized for Marinescu's alg. B and Papa.& Sahgal's 37N test.

A slightly disappointing performance displays Marinescu's algorithm A, which corresponds with Suk & Reddy's test A. The score of this pattern remains 10% below of what might be expected. A clear explanation is still lacking, but a possible origin could be the insufficient number of read operations that leads to a bad detection of the data retention faults.
On the contrary, the Marching 0's/1's pattern shows an excellent detection capability, despite of its bit line following addressing order. The main difference with the comparable MATS-plus pattern is the fact that each write operation is succeeded with a read action.
Another possible explanation for this difference would be the presence of write recovery failures, i.e. when a write action is immediately followed by a read action of the same memory location. However, when the address changes in the mean time, the effect was not observed. Further examinations are needed to determine the real origin.

The same trends are visible when the patterns with the addressing variations were executed using the tightened address access timing diagram, see appendix C.
Although at first sight the C- and E-versions give the best results at the nearly good devices, the B- and D-versions displayed their advantages again at the more serious damaged memories, see plot 3 in appendix F.
However, this was caused by a few memories suffering from address access problems. The origin has been established with the aid of the GALTCOL patterns, which showed an extreme sensitivity to the address access problem.

In fact, the G-version was the only addressing variation for which the tightened timing conditions resulted in a consistent better performance.

Once more, the fault model has shown its considerable impact on the fault coverage at the devices with coupling faults in the decoder and read/write logic, see the next section.

5.4.2 Discussion of the Permanent Faults

The main part of test program 2 concerned a more detailed examination of the pattern behaviour in case of permanent faults. The outcomes of test program 1 fastened the attention to the procedures based on the stuck-at and the coupling fault model.
Several addressing orders and data backgrounds have been used to investigate their possible influences on the detection properties of the patterns.

The addressing orders were already mentioned in the preceding section.

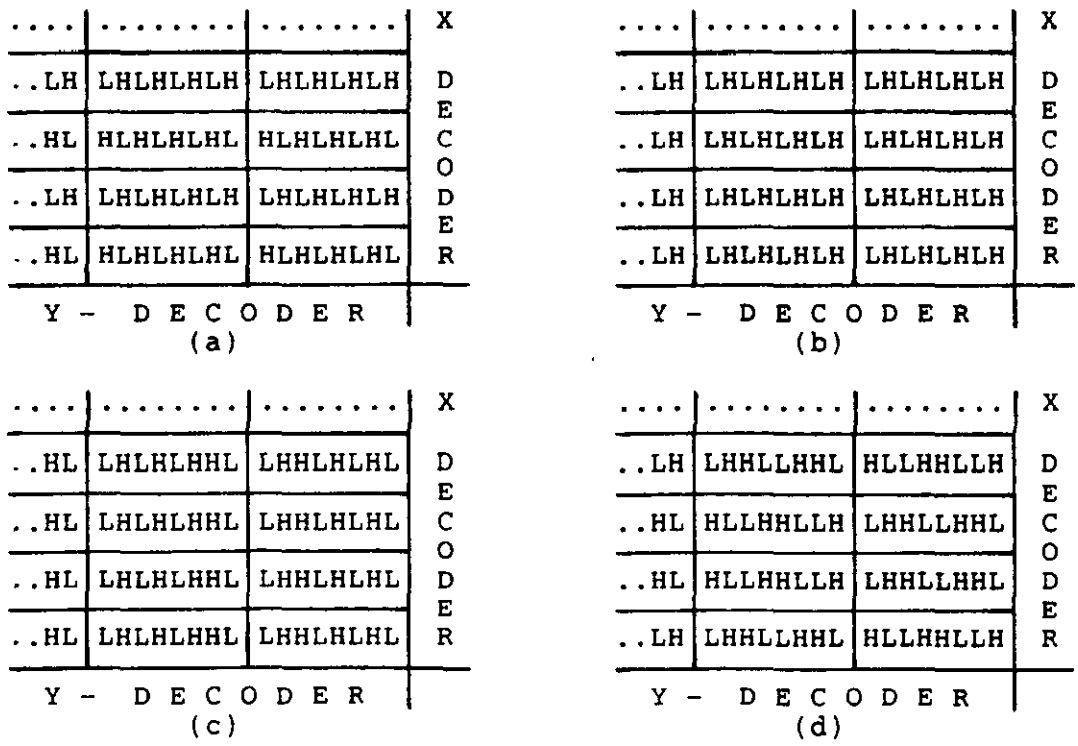


Fig.33 In one array half the bit line settings due to the (a) topological checkerboard, (b) topological 0's, (c) logical 0's and (d) cell-based 0's.

All patterns have been used to examine the impact of the data backgrounds on the fault coverage. The set was executed four times, namely with a topological 0/1 background, a topological checkerboard background, a logical 0/1 background and a data structure based on the physical position of the individual memory cells, see figure 33.

The topological backgrounds have the effect that the bit lines of the memory array are alternately set in a high-low order. This is restricted to the columns for the 0/1 background, see figure 33(b), and also extended to the rows for the checkerboard situation, see figure 33(a).
On the contrary, the logical data setting takes only the external pin configuration into account and places all the eight data pins in the 0 or the 1 position, see figure 33(c) for the final bit line setting of the array.
The fourth data background has been developed to examine the potential effect of parasitic coupling between neighbouring cells by means of a worst case data distribution, see figure 33(d).
The access of whole nibbles instead of single cells is also indicated by means of the rectangles in figure 33.

The test program resulted to the following observations:

1. The gains of a better fault model and differences due the data settings can be visualized if the device contains serious failures in the decoder or the read/write logic. On the contrary, the influence of the addressing sequence can be neglected in these cases;

2. The ad-hoc test patterns show again their uncertain behaviour at this wafer, including the existing production pattern set.

ad.1 The shape of the average pattern set performances changes considerably when the number of detected faults has increased, compare plot 1 and 2 in appendix F.
On the one hand, the Marching 0's/1's and the MATS-plus patterns have become less effective due to the limitations of the fault model, whereas the procedures based on the coupling fault model show improved scoring rates, see plot 2.
On the other hand, the addressing order doesn't play an important role anymore. The mutual differences between the patterns within the second category are rather small, see plot 2.
Hence, the properties of the employed fault model become again visible at the memories with large fault numbers. It doesn't make sense to use the 30N or 37N test, while the more efficient patterns like Marinescu's algorithm C, 11N memory operations, have a comparable detection capacity, see plot 2.

The choice of the data background has a considerable impact on the fault coverage for the memory category with large fault numbers. This results into performance differences up to 30%, see plot 2 in appendix F.
Remarkable figures caused by the fact that the logical 0's/1's and the cell-based data settings do not place all neighbouring bit lines of the memory array in an alternating high-low order, as the topological data settings do, see figure 33. That makes the other two data backgrounds inferior to detect all coupling faults in the memory array, especially between cells of the same nibble.
The impact of the data backgrounds is less important at the nearly good devices. The patterns detect somewhat more faults when the topological data backgrounds are used in comparision with the other two data settings, but the differences are restricted to about 5%, thus small.

One aspect hasn't yet been investigated, namely a data background which also places the lines of the internal data bus in an alternating high-low order. It might be an interesting topic for further research.

2. The ad-hoc test patterns show again their uncertain behaviour, including the existing production pattern set.

ad.2 First, the ad-hoc procedures DIACOL, DIAPAT, Surround Disturb and API test 2a are again highly disappointing for the memories with considerable number of malfunctions, see plot 2 in appendix F.
Secondly, the GALTCOL patterns showed again their exceptional sensitivity for the address access problem.
Thirdly, the Marching 0's/1's algorithm displayed remarkable detection rates at the nearly good devices, but its coverage fell back when the fault numbers increased, compare plot 1 and 2.
Hence, the ad-hoc procedures showed the same uncertain behaviour as we saw with test program 1. Above, the Surround Disturb performance was again of less importance and asks once more for a reconsideration of the existing pattern set.

5.5 Summary and Memory Test Strategies

5.5.1 Summary of the Test Results

The location of time-dependent faults and design errors appears to be a difficult task. However, the execution of a large pattern collection has proven its advantages. Remarkable scores of some procedures point to particular problems and quicken the tracing of the failure origins.
So, in combination with the use of average pattern set performances and other distributions, a very effective set of analysis tools becomes available for the engineering environment. The plots may be generated in an easy way with the aid of software programs like RS/1 [51].

The 2-coupling fault model proves to be the most suitable method to test these static memories for permanent faults, especially in case of serious decoder failures and malfunctions of the read/write logic.
The properties are valid to all the procedures of the 2-coupling category. The results affirm not only the ideas of S.M. Thatte, but affirm the assumptions made by Marinescu and Suk & Reddy too, i.e. the simultaneous presence of various failure types won't affect the final detection properties. So, the more efficient 2-coupling patterns are preferred. Particularly Marinescu's algorithm C seems promising which requires only 11N memory accesses.
The ad-hoc data patterns as well as algorithms based on the stuck-at model or the pattern-sensitive approach display a minor detection capability.
One might expect that these results have a general meaning, because the trends were visible under several timing conditions and at all the four wafers.

When the memories contain a considerable number of malfunctions the addressing order plays a minor role. On the contrary, the patterns show a divergent behaviour at the nearly good devices. Just then the addressing sequence seems the most important parameter.
On the one hand, the data retention failures are mainly detected by the patterns with a word line following addressing sequence. On the other hand, patterns show an increased sensitivity for the address access problem when the address generation follows the bit lines. The results of the other address variations were not convincing, even if the timing was tightened.
Opposite, the average pattern set performance for permanent faults displays a remarkable similarity at these nearly good memories. No dominating factor can be recognized like a preferable fault model or a special read/write order.
Above, the Marching 0's/1's procedure showed excellent scores, whereas Marinescu's algorithm A disappointed a little. A convincing explanation of this phenomenon is still lacking. The differences between both patterns point

to the possibility of write recovery failures, but even data retention faults can not be excluded.

The impact of the data background on the fault coverage of the patterns proves to be considerable. Especially coupling faults between neighbouring memory cells sharing the same nibble, remain undetected when the bit lines have not been set in an alternating high-low order.

The existing production pattern set asks for improvements. The detection properties are mainly provided by the Marching 0's/1's pattern, whereas the contribution of the Surround Disturb procedure appeared to be inferior.

The following two questions are not yet answered:

1. Can Marinescu's algorithm C be extended with a few read operations in such a way that the data retention and/or write recovery failures are better detected ?

2. Will a data background improve the fault coverage when the lines of the internal data bus are also tested for coupling faults ?

5.5.2 Memory Test Strategies

When one has the disposal of a large pattern collection and some statistical tools as well, it has been proven that both device as wafer characterization can be improved significantly. Since patterns are known with a specific fault detection sensitivity, high performance scores of particular patterns will provide useful information for further analysis. These conclusions have especially significance for the engineering environment.

The most suitable approach to memory testing for incoming inspection is less obvious.

First, if only the logical function of a memory has to be tested one of the 2-coupling procedures may be preferred. The results have shown that the addressing order can be randomly chosen. However, internal data inversions may have a large impact on the pattern detection properties, especially in case of a memory with nibble or byte access. The pattern-sensitive algorithms even require knowledge about the possible internal address scrambling.

Secondly, time-dependent faults are hard to test. These faults are very device specific and testing demands detailed information of the internal device structure. The execution of a large pattern set may help too.

6 CONCLUSIONS AND SUGGESTIONS FOR FURTHER RESEARCH

6.1 Conclusions

We recapitulate the content of this thesis by the following seven main points;

1. Functional testing of RAMs can be improved with the aid of fault models. The advantages are plural.
   First, the fault coverage increases when algorithms are used based on a well-defined fault model.
   Secondly, a considerable reduction of the test time is possible, since the optimized patterns require a substancially fewer number of memory accesses.
   Thirdly, one has the guarantee that the specified faults will be detected.

2. The fault coverage of the stuck-at model turns out to be too limited if the memories contain hardly detectable coupling failures, especially caused by faults in the decoder or read/write logic.
   The test procedures for pattern-sensitive faults were disappointing. The result isn't surprising because a basically static memory has been used as test vehicle with nibble access.
   Both test programs showed the strenght of the 2-coupling fault model, based on the fault translation principles of S.M. Thatte. However, the results were obtained with only one device type, in this case Philips' 16K static RAM. The proves become more general if the same conclusions can be drawn at other memories, for instance a dynamic type.

3. The ad-hoc procedures like GALPAT, Shifted Diagonals, etc. show a decreased performance if a fault doesn't suit to their specific detection sensitivity. The patterns display a very uncertain behaviour and require an unnecessarily large amount of memory operations. Only in case of special failures, mostly time-dependent, some procedures showed an increased performance.

4. The addressing order doesn't influence the properties of the patterns for permanent faults. However, the detection of time-dependent failures might be improved with a specific addressing sequence but requires knowledge about the internal memory operations.

5. The employed data background may have a considerable impact on the fault coverage. The procedures displayed a decreased performance at the nibble oriented Philips memory when the bit lines were not set in an alternating high-low order. This causes that some coupling faults between neighbouring memory cells remain undetected.

6.

Although the Marching 0's/1's procedure displayed an excellent performance at the nearly good devices, a drop was viewed at the memories with coupling faults due to the limitations of the fault model.

7. Design errors and processing shortcomings result frequently into time-dependent faults. The tracing of these failure origins is mostly a very difficult task on account of their unexpected and specific nature. The execution of a large pattern set proves to be an important help by which the analyse process might be improved.

6.2 Suggestions for Further Research

The outcomes of test program 2 point to the possibility that in some cases single data retention and/or write recovery failures remain undetected by most patterns.
The indication is based on the excellent detection rates of the Marching 0's/1's pattern at the memories with a few faults, whereas Marinescu's algorithm A showed a little disappointing performance. More detailed investigations are necessary to establish the origin, i.e. to decide whether the effect has a general meaning or caused by this specific device.
Marinescu's algorithm C might be improved in this way through the addition of some read operations. The resulting pattern should give a better overall fault coverage with nearly the same number of memory operations as the Marching 0's/1's procedure.

Although testing of embedded RAMs becomes more and more a problem, time was lacking to do more detailed research in this field. However, these investigations mapped out the differences between the test methods and showed that even simple march patterns will satisfy. This opens the possibility to proceed in the direction of self-test or start some other experiments with embedded RAMs.

Eindhoven, June 1986
Philips Research Laboratories,
Nederlandse Philips Bedrijven B.V.

ACKNOWLEDGEMENTS

I thank Frans Beenker, Jan Koomen and Rinus van Weert for their useful advices and critisism. My special thanks go to Dick Kleinloog and Gert-Jan Roskam of the group "Test and Analysis" and others for their indispensable help by the implementation of the test programs and interpretation of the results.
I thank too the members of the group "CAD for VLSI systems" for their hospitality and companionship, my girlfriend Jikke and my parents for their patience, and last but not least Texas Instruments for making such indestructible pocket calculators.

REFERENCES

[1] Breuer, M.A. and A.D. Friedman
Diagnosis & reliable design of digital systems.
Woodland Hills, Cal.: Computer Science Press, 1976 and London: Pitman, 1977.
Digital system design series

[2] Muehldorf, E.I. and A.D. Savkar
LSI logic testing - an overview.
IEEE Trans. Comput., Vol. C-30(1981), p. 1-17.

[3] Ligthart, M. and R. Stans
A fault model for PLAs.
Internal Philips report, April 1986. To be published.

[4] Ostapko, D.L. and S.J. Hong
Fault analysis and test generation for Programmable Logic Arrays (PLA's).
IEEE Trans. Comput., Vol. C-28(1979), p. 617-627.

[5] Williams, T.W. and K.P. Parker
Design for testability - a survey.
Proc. IEEE, Vol. 71(1983), p. 98-112.

[6] Fujiwara, H.
Logic testing and design for testability.
Cambridge, Mass.: MIT Press, 1985.
MIT Press series in computer systems. P. 213-237, 259-266.

[7] Koenemann, B.
Built-in testing: State of the art.
In: Curriculum for test technology, Proc. Workshop, Minneapolis, 16-17 Nov. 1983.
New York: IEEE, 1983. P. 83-89.

[8] Beenker, F.P.M. et al.
Macro testing: A VLSI system test approach.
Accepted for publication in: IEEE Design & Test Comput., Vol. 3(1986) or Vol. 4(1987).

[9] McCluskey, E.J. and S. Bozorgui-Nesbat
Design for autonomous test.
In: Digest of papers 11th Test Conf., Philadelphia, 11-13 Nov. 1980.
New York: IEEE, 1980. P. 15-21.

[10] Eichelberger, E.B. and E. Lindbloom
Random-pattern coverage enhancement and diagnosis for LSSD logic self-test.
IBM J. Res. & Develop., Vol. 27(1983), p. 265-272.

[11] De Man, H.
CAD tools for third generation custom VLSI design.
In: ESSCIRC '85. Proc. 11th European Solid State Circuits Conf., Toulouse, 16-18 Sept. 1985.
Toulouse: Université Paul Sabatier, 1985. P. 256-256c.

[12] Guterl, F. et al.
The one-month chip.
IEEE Spectrum, Vol. 21, No. 9(Sept. 1984), p. 28-49. Correction: Vol. 21, No. 12(Dec. 1984), p. 16.

[13] Prince, B. and G. Due-Gundersen
Semiconductor memories.
Chichester: Wiley, 1983. P. 155-158.

[14] Lo, T.C. et al.
A 64K fet dynamic random access memory: Design considerations and description.
IBM J. Res. & Develop., Vol. 24(1980), p. 318-327.

[15] Philips Research Laboratories, Eindhoven; Documentation K16-SRAM CMOS V3-V4 AMDC
Nat. Lab., date 10-10-85.

[16] Stapper, C.H.
Yield model for 256K RAMs and beyond.
In: Digest of technical papers IEEE Int. Solid-State Circuits Conf. (ISSCC), San Francisco,
10-12 Febr. 1982. Ed. by L. Winner. New York: IEEE, 1982. P. 12-13.

[17] Kokkonen, K. et al.
Redundancy techniques for fast static RAMs.
In: Digest of technical papers IEEE Int. Solid-State Circuits Conf. (ISSCC), New York,
18-20 Febr. 1981. Ed. by L. Winner. New York: IEEE, 1981. P. 80-81.

[18] Smith, R.J. et al.
32K and 16K static MOS RAMs using laser redundancy techniques.
In: Digest of technical papers IEEE Int. Solid-State Circuits Conf. (ISSCC), San Francisco,
10-12 Febr. 1982. Ed. by L. Winner. New York: IEEE, 1982. P. 252-253.

REFERENCES

[19] Bindels, J.F.M. et al.
Cost-effective yield improvement in fault-tolerant VLSI memory.
In: Digest of technical papers IEEE Int. Solid-State Circuits Conf. (ISSCC), New York, 18-20 Febr. 1981. Ed. by L. Winner. New York: IEEE, 1981. P. 82-83.

[20] Kirschner, N.
An interactive descrambler program for RAMs with redundancy.
In: Digest of papers 13th Int. Test Conf., Philadelphia, 15-18 Nov. 1982.
New York: IEEE, 1982. P. 252-257.

[21] Persoon, E.H.J. and C.J.B. Vandenbulcke
Digital audio: Examples of the application of the ASP integrated signal processor.
Philips Tech. Rev., Vol. 42(1986), p. 201-216. Dutch version: Philips Tech. Tijdschr., Vol. 42(1985), p. 210-226.

[22] Kuban, J. and B. Bruce
The MC6804P2 built-in self-test.
In: Proc. 14th Int. Test Conf., Philadelphia, 18-20 Oct. 1983.
New York: IEEE, 1983. P. 295-300.

[23] Intelligent peripheral controller. Documentation Motorola MC68120/MC68121.

[24] Abadir, M.S. and H.K. Reghbati
Functional testing of semiconductor random access memories.
Comput. Surv., Vol. 15, No. 3(Sept. 1983), p. 175-198.

[25] Barraclough, W. et al.
Techniques for testing the microcomputer family.
Proc. IEEE, Vol. 64(1976), p. 943-950.

[26] Knaizuk, Jr., J. and C.R.P. Hartmann
An algorithm for testing random access memories.
IEEE Trans. Comput., Vol. C-26(1977), p. 414-416.

[27] Knaizuk, Jr., J. and C.R.P. Hartmann
An optimal algorithm for testing stuck-at faults in random access memories.
IEEE Trans. Comput., Vol. C-26(1977), p. 1141-1144.

[28] Nair, R.
Comments on "An optimal algorithm for testing stuck-at faults in random access memories".
IEEE Trans. Comput., Vol. C-28(1979), p. 258-261.

[29] Thatte, S.M. and J.A. Abraham
Testing of semiconductor random access memories.
In: Proc. 7th Annual Int. Conf. on Fault-Tolerant Computing, Los Angeles, 28-30 June 1977.
New York: IEEE, 1977. P. 81-87.

[30] Hayes, J.P.
Testing memories for single-cell pattern-sensitive faults.
IEEE Trans. Comput., Vol. C-29(1980), p. 249-254.

[31] Courtois, B. and H. Sahami
Structural testing of NMOS RAMs.
St. Martin d'Heres (France): IMAG/TIM3 Lab., March 1984.
Research report RR432.

[32] Varma, P. et al.
On-chip testing of embedded RAMs.
In: Proc. Custom Integrated Circuits Conf., Rochester, N.Y., 21-23 May 1984.
New York: IEEE, 1984. P. 286-290.

[33] Schwehr, J.B. et al.
Electrical characterization of 16K static RAMs.
Griffiss Air Force Base, N.Y.: Rome Air Development Center, 1982.
Technical report No. RADC-TR-82-40. P. 72-98. NTIS Order No. AD-A116 111/6.

[34] Bardell, Jr., P.H. and W.H. McAnney
Self-test of random access memories.
In: Proc. 16th Int. Test Conf., Philadelphia, 19-21 Nov. 1985.
New York: IEEE, 1985. P. 352-355.

[35] Sun, Z. and L.-T. Wang
Self-testing of embedded RAMs.
In: Proc. 15th Int. Test Conf., Philadelphia, 16-18 Oct. 1984.
New York: IEEE, 1984. P. 148-156

[36] Nair, R. et al.
Efficient algorithms for testing semiconductor random-access memories.
IEEE Trans. Comput., Vol. C-27(1978), p. 572-576.

[37] Suk, D.S. and S.M. Reddy
A march test for functional faults in semiconductor random access memories.
IEEE Trans. Comput., Vol. C-30(1981), p. 982-985.

[38] Marinescu, M.
Simple and efficient algorithms for functional RAM testing.
In: Digest of papers 13th Int. Test Conf., Philadelphia, 15-18 Nov. 1982.
New York: IEEE, 1982. P. 236-239.

[39] Papachristou, C.A. and N.B. Sahgal
An improved method for detecting functional faults in semiconductor random access memories.
IEEE Trans. Comput., Vol. C-34(1985), p. 110-116.

[40] Hayes, J.P.
Detection of pattern-sensitive faults in random-access memories.
IEEE TRans. Comput., Vol. C-24(1975), p. 150-157.

[41] Scrini, V.P.
API tests for RAM chips.
Computer, Vol. 10, No. 7(July 1977), p. 32-35.

[42] Suk, D.S. and S.M. Reddy
Test procedures for a class of pattern-sensitive faults in semiconductor random-access memories.
IEEE Trans. Comput., Vol. C-29(1980), p. 419-429.

[43] Saluja, K.K. and K. Kinoshita
Test pattern generation for API faults in RAM.
IEEE Trans. Comput., Vol. C-34(1985), p. 284-287.

[44] You, Y. and J.P. Hayes
A self-testing dynamic RAM chip.
In: Proc. Conf. on Advanced Research in VLSI, Cambridge, Mass., 23-25 Jan. 1984.
Ed. by P. Penfield.
Dedham, Mass.: Artech House, 1984. P. 159-168.

[45] Westcott, D.
The self-assist test approach to embedded arrays.
In: Digest of papers 12th Int. Test Conf., Philadelphia, 27-29 Oct. 1981.
New York: IEEE, 1981. P. 203-207.

[46] Saluja, K.K. and Kim Thang Le
Testable design of large random access memories.
Integration VLSI J., Vol. 2(1984), p. 309-330.

[47] Kinoshita, K. and K.K. Saluja
Built-in testing of memory using on-chip compact testing scheme.
In: Proc. 15th Int. Test Conf., Philadelphia, 16-18 Oct. 1984.
New York: IEEE, 1984. P. 271-281.

[48] Nicolaidis, M.
An efficient built-in self-test scheme for functional test of embedded RAMs.
In: Digest of papers 15th Annual Symp. on Fault-Tolerant Computing, Ann Arbor, Mich., 19-21 June 1985.
New York: IEEE, 1985. P. 118-123.

[49] Kuban, J.R. and J.E. Salick
Testing approaches in the MC68020.
VLSI Design, Vol. 5, No. 11(Nov. 1984), p. 22-30.

[50] J389 programming manual. Documentation Teradyne. May 1984. Section 2.8 and 2.9.

[51] Syllabus of the RS/1 Video Course.
BBN Software Products Co., A Subsidiary of Bolt, Beranek & Newman, Inc., 10 Fawcett Street, Cambridge, MA 02138, U.S.A.

APPENDIX A : Descriptions of some test algorithms

Descriptions of several algorithms discussed in chapter 3 are given below. The numbering of the algoritms refers to that of table 2. Memories are assumed with N addresses and a bit wide data mechanism. The following notations will be used for the descriptions of the patterns:

a. a read of a cell and expecting the logical 0-value, will be denoted by R(0);
b. similarly, R(1) is used to indicate a read of a cell containing the logical 1-value;
c. a write of a cell with the logical 0-value will be denoted by the abbreviation W(0);
d. likewise, W(1) stands for a write operation with the 1-value.

1. MSCAN (section 3.4, 4N address operations, stuck-at category)

```
begin
      1.for i=0 to N-1 do W(0) in cell[i] od;
      2.for i=0 to N-1 do R(0) in cell[i] od;
      3.for i=0 to N-1 do W(1) in cell[i] od;
      4.for i=0 to N-1 do R(1) in cell[i] od
end.
```

or:

addr.	step1	step2	step3	step4
0	W(0)	R(0)	W(1)	R(1)
1	W(0)	R(0)	W(1)	R(1)
:	:	:	:	:
N-1	W(0)	R(0)	W(1)	R(1)

2. A.T.S. (section 3.4, 4N operations, stuck-ats, for a memory with a non-creative decoder)

```
p(0) := { 0 <= i < N | i = 0 (modulo 3) }
p(1) := { 0 <= i < N | i = 1 (modulo 3) }
p(2) := { 0 <= i < N | i = 2 (modulo 3) }

begin
      1.for all i { p(1) do W(0) in cell[i] od;
        for all i { p(2) do W(0) in cell[i] od;

      2.for all i { p(0) do W(1) in cell[i] od;

      3.for all i { p(1) do R(0) in cell[i] od;

      4.for all i { p(1) do W(1) in cell[i] od;
```

```
      5.for all i { p(2) do R(0) in cell[i] od;

      6.for all i { p(0) do R(1) in cell[i] od;
        for all i { p(1) do R(1) in cell[i] od;

      7.for all i { p(0) do W(0) in cell[i];
                           R(0) in cell[i] od;

      8.for all i { p(2) do W(1) in cell[i];
                           R(1) in cell[i] od
end.
```

or:

part	step1	step2	step3	step4	step5	step6	step7	step8
p(0)		W(1) : W(1)				R(1) : R(1)	W(0)R(0) : W(0)R(0)	
p(1)	W(0) : W(0)		R(0) : R(0)	W(1) : W(1)		R(1) : R(1)		
p(2)	W(0) : W(0)				R(0) : R(0)			W(1)R(1) : W(1)R(1)

3. MARCHING 0's/1's (section 3.4 and 3.5.2,
14N operations, stuck-at and 2-coupling category)

```
begin
      1.for i=0 to N-1 do W(0) in cell[i] od;
      2.for i=0 to N-1 do R(0) in cell[i];
                          W(1) in cell[i];
                          R(1) in cell[i] od;
      3.for i=0 to N-1 do R(1) in cell[i];
                          W(0) in cell[i];
                          R(0) in cell[i] od;
      4.step 1-3 are repeated with complemented data
end.
```

4.a M.A.T.S.-or (section 3.4, 4N address operations,
stuck-at for a memory with a logical wired-or behaviour)

```
begin
      1.for i=0 to N-1 do W(0) in cell[i] od;
      2.for i=0 to N-1 do R(0) in cell[i];
                          W(1) in cell[i] od;
      3.for i=0 to N-1 do R(1) in cell[i] od
end.
```

or:

addr.	step1	step2	step3
0	W(0)	R(0)W(1)	R(1)
1	W(0)	R(0)W(1)	R(1)
:	:	:	:
N-1	W(0)	R(0)W(1)	R(1)

4.b M.A.T.S.-and (section 3.4, 4N operations,
stuck-at for a memory with a logical wired-and behaviour)

```
begin
      1.for i=0 to N-1 do W(1) in cell[i] od;
      2.for i=0 to N-1 do R(1) in cell[i];
                          W(0) in cell[i] od;
      3.for i=0 to N-1 do R(0) in cell[i] od
end.
```

or:

addr.	step1	step2	step3
0	W(1)	R(1)W(0)	R(0)
1	W(1)	R(1)W(0)	R(0)
:	:	:	:
N-1	W(1)	R(1)W(0)	R(0)

4.c MATS-plus, see appendix F (section 3.4, 5N operations,
stuck-ats for a memory with an arbitrary logical behaviour)

6. MASEST (table 2, 10N address operations,
2-coupling category)

```
begin
   1.for i=0 to N-1 do write alternating data in cell[i] od;
   2.for i=0 to N-1 do read data in cell[i];
                       read opposite data in cell[N-1-i];
                       read data in cell[i] od;
   3.for i=0 to N-1 do read data in cell[i] od;
   4.step 1-3 are repeated using complemented data
end.
```

8. SHIFTED DIAGONAL (table 2, $2N^{3/2}$ + 6N address operations,
2-coupling category)

```
begin
      1.write 0's in all cells, except the cells of a test
        diagonal which are written with 1's;
      2.read all memory cells column for column;
      3.shift the diagonal one position to the right;
      4.repeat step 2 and 3 until all columns are sequenced;
      5.repeat step 1-4 using complemented data
end.
```

9. ALGORITHM C (section 3.5.3, 11N address operations, 2-coupling category, proposed by Marinescu)

```
begin
      1.for i=0 to N-1 do W(0) in cell[i] od;
      2.for i=0 to N-1 do R(0) in cell[i];
                          W(1) in cell[i] od;
      3.for i=0 to N-1 do R(1) in cell[i];
                          W(0) in cell[i] od;
      4.for i=0 to N-1 do R(0) in cell[i] od;
      5.for i=N-1 to 0 do R(0) in cell[i];
                          W(1) in cell[i] od;
      6.for i=N-1 to 0 do R(1) in cell[i];
                          W(0) in cell[i] od;
      7.for i=N-1 to 0 do R(0) in cell[i] od
end.
```

or:

addr.	step1	step2	step3	step4
0	W(0)	R(0)W(1)	R(1)W(0)	R(0)
1	W(0)	R(0)W(1)	R(1)W(0)	R(0)
:	:	:	:	:
N-1	W(0)	R(0)W(1)	R(1)W(0)	R(0)

addr.	step5	step6	step7
0	R(0)W(1)	R(1)W(0)	R(0)
:	:	:	:
N-2	R(0)W(1)	R(1)W(0)	R(0)
n-1	R(0)W(1)	R(1)W(0)	R(0)

11. ALGORITHM A (section 3.5.3, 15N address operations, 2-coupling category proposed by Marinescu)

```
begin
      1.for i=0 to N-1 do W(0) in cell[i] od;
      2.for i=0 to N-1 do R(0) in cell[i];
                          W(1) in cell[i];
                          W(0) in cell[i];
                          W(1) in cell[i] od;
      3.for i=0 to N-1 do R(1) in cell[i];
                          W(0) in cell[i];
                          W(1) in cell[i] od;
      4.for i=N-1 to 0 do R(1) in cell[i];
                          W(0) in cell[i];
                          W(1) in cell[i];
                          W(0) in cell[i] od;
      5.for i=N-1 to 0 do R(0) in cell[i];
                          W(1) in cell[i];
                          W(0) in cell[i] od
end.
```

or:

addr.	step1	step2	step3
0	W(0)	R(0)W(1)W(0)W(1)	R(1)W(0)W(1)
1	W(0)	R(0)W(1)W(0)W(1)	R(1)W(0)W(1)
:	:	:	:
N-1	W(0)	R(0)W(1)W(0)W(1)	R(1)W(0)W(1)

addr.	step4	step5
0	R(1)W(0)W(1)W(0)	R(0)W(1)W(0)
:	|	|
N-2	R(1)W(0)W(1)W(0)	R(0)W(1)W(0)
N-1	R(1)W(0)W(1)W(0)	R(0)W(1)W(0)

12. TEST B (section 3.5.3, 16N address operations,
2-coupling category proposed by Suk and Reddy)

```
begin
      1.for i=0 to N-1 do W(0) in cell[i] od;
      2.for i=0 to N-1 do R(0) in cell[i];
                          W(1) in cell[i];
                          R(1) in cell[i];
                          W(0) in cell[i];
                          R(0) in cell[i];
                          W(1) in cell[i] od;
      3.for i=0 to N-1 do R(1) in cell[i];
                          W(0) in cell[i];
                          W(1) in cell[i] od;
      4.for i=N-1 to 0 do R(1) in cell[i];
                          W(0) in cell[i];
                          W(1) in cell[i];
                          W(0) in cell[i] od;
      5.for i=N-1 to 0 do R(0) in cell[i];
                          W(1) in cell[i];
                          W(0) in cell[i] od
end.
```

addr.	step2	step3
0	R(0)W(1)R(1)W(0)R(0)W(1)	R(1)W(0)W(1)
1	R(0)W(1)R(1)W(0)R(0)W(1)	R(1)W(0)W(1)
:	:	:
N-1	R(0)W(1)R(1)W(0)R(0)W(1)	R(1)W(0)W(1)

addr.	step4	step5
0	R(1)W(0)W(1)W(0)	R(0)W(1)W(0)
:	|	|
N-2	R(1)W(0)W(1)W(0)	R(0)W(1)W(0)
N-1	R(1)W(0)W(1)W(0)	R(0)W(1)W(0)

14. WALKING 0's/1's (section 3.5.2, $2N^2 + 8N$ operations, 2-coupling category)

```
begin
      1.for i=0 to N-1 do W(0) in cell[i]
                       od;
      2.for j=0 to N-1
                do W(1) in cell[j];
                   R(1) in cell[j];
                   for i=0 to i=N-1 and i≠j
                            do R(0) in cell[i]
                            od;
                   R(1) in cell[j];
                   W(0) in cell[j]
                od;
      3.repeat 1 and 2 with complemented data
end.
```

15. GALPAT (section 3.5.2, $4N^2 + 4N$ operations, 2-coupling category)

```
begin
      1.for i=0 to N-1 do W(0) in cell[i]
                       od;
      2.for j=0 to N-1
                do W(1) in cell[j];
                   for i=0 to i=N-1 and i≠j
                            do R(1) in cell[j];
                               R(0) in cell[i]
                            od;
                   R(1) in cell[j];
                   W(0) in cell[j]
                od;
      3.repeat 1 and 2 with complemented data
end.
```

16. GALTCOL (section 3.5.2, $4N^{3/2} + 4N$ operations, 2-coupling category)

```
begin
      1.for i=0 to N-1 do W(0) in cell[i]
                       od;
      2.for j=0 to N-1
                do W(1) in cell[j];
                   for i=0 to i="end of column" and i≠j
                            do R(1) in cell[j];
                               R(0) in cell[i]
                            od;
                   R(1) in cell[j];
                   W(0) in cell[j] od;
      3.repeat 1 and 2 with complemented data
end.
```

18.a TEST A (section 3.5.3, 30N address operations,
2-coupling category proposed by Nair et.al.)

addr.	step1	step2	step3	step4	step5
0	W(0)	R(0)W(1)	R(1)	R(1)W(0)	R(0)
1	W(0)	R(0)W(1)	R(1)	R(1)W(0)	R(0)
:	:	:	:	:	:
N-2	W(0)	R(0)W(1)	R(1)	R(1)W(0)	R(0)
N-1	W(0)	R(0)W(1)	R(1)	R(1)W(0)	R(0)

step6	step7	step8	step9	step10
R(0)W(1)	R(1)	R(1)W(0)	R(0)	R(0)W(1)W(0)
R(0)W(1)	R(1)	R(1)W(0)	R(0)	R(0)W(1)W(0)
:	:	:	:	:
R(0)W(1)	R(1)	R(1)W(0)	R(0)	R(0)W(1)W(0)
R(0)W(1)	R(1)	R(1)W(0)	R(0)	R(0)W(1)W(0)

step11	step12	step13	step14	step15
R(0)	R(0)W(1)W(0)	R(0)	W(1)	R(1)W(0)W(1)
R(0)	R(0)W(1)W(0)	R(0)	W(1)	R(1)W(0)W(1)
:	:	:	:	:
R(0)	R(0)W(1)W(0)	R(0)	W(1)	R(1)W(0)W(1)
R(0)	R(0)W(1)W(0)	R(0)	W(1)	R(1)W(0)W(1)

step16	step17	step18
R(1)	R(1)W(0)W(1)	R(1)
R(1)	R(1)W(0)W(1)	R(1)
:	:	:
R(1)	R(1)W(0)W(1)	R(1)
R(1)	R(1)W(0)W(1)	R(1)

18.b 37N TEST (section 3.5.3, 37N address operations, 2-coupling category proposed by Papachristou and Sahgal)

add	step1	step2	step3	step4
0	W(0)	R(0)W(1)R(1)	R(1)W(0)R(0)	R(0)W(1)W(0)
1	W(0)	R(0)W(1)R(1)	R(1)W(0)R(0)	R(0)W(1)W(0)
:	:	:	:	:
N-2	W(0)	R(0)W(1)R(1)	R(1)W(0)R(0)	R(0)W(1)W(0)
N-1	W(0)	R(0)W(1)R(1)	R(1)W(0)R(0)	R(0)W(1)W(0)

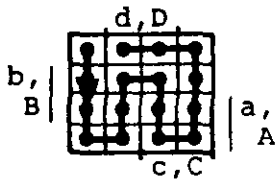
step5	step6	step7	step8
R(0)W(1)	R(1)W(0)W(1)	R(1)W(0)	R(0)W(1)W(0)
R(0)W(1)	R(1)W(0)W(1)	R(1)W(0)	R(0)W(1)W(0)
:	:	:	:
R(0)W(1)	R(1)W(0)W(1)	R(1)W(0)	R(0)W(1)W(0)
R(0)W(1)	R(1)W(0)W(1)	R(1)W(0)	R(0)W(1)W(0)

step9	step10	step11	step12
R(0)W(1)	R(1)W(0)	R(0)W(1)W(0)	R(0)W(1)
R(0)W(1)	R(1)W(0)	R(0)W(1)W(0)	R(0)W(1)
:	:	:	:
R(0)W(1)	R(1)W(0)	R(0)W(1)W(0)	R(0)W(1)
R(0)W(1)	R(1)W(0)	R(0)W(1)W(0)	R(0)W(1)

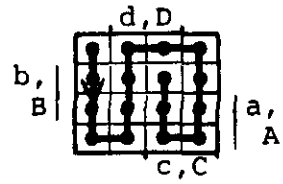
step13	step14	step15
R(1)W(0)W(1)	R(1)W(0)	R(0)W(1)W(0)
R(1)W(0)W(1)	R(1)W(0)	R(0)W(1)W(0)
:	:	:
R(1)W(0)W(1)	R(1)W(0)	R(0)W(1)W(0)
R(1)W(0)W(1)	R(1)W(0)	R(0)W(1)W(0)

24. A.P.I. test 2 (section 3.6.5, 41N address operations, pattern-sensitive category proposed by Saluja and Kinoshita)

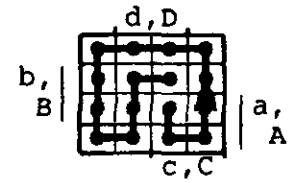
The following three rather arbitrary Hamiltonian paths are used in the algorithm. According to the tiling method of figure 13, each dot in the Karnaugh maps represents the logical state of the four neighborhood cells, depending on the present mode.



Hamiltonian H1



Hamiltonian H2



Hamiltonian H3

begin

1. initialize the whole memory in the all 0-state;

mode 1:

2. perform a R(0) in all the t-base cells;
3. sequence the fifteen state transitions according to path H1 in the A,B,C,D-neighborhood cells and perform a R(0) in all the t-base cells after each state transition;
4. bring the A,B,C,D-neighborhood cells in the all 0-state with an additional W(0) of the D-cells;

mode 2:

5. perform a R(0) in all the T-base cells;
6. sequence the fifteen state transitions according to path H2 in the a,b,c,d-neighborhood cells and perform a R(0) in all the T-base cells after each state transition;
7. bring the a,b,c,d-neighborhood cells in the all 1-state with an additional W(1) of the a-cells;

mode 1:

8. perform a R(1) in all the t-base cells;
9. sequence the fifteen state transitions according to path H2 in the A,B,C,D-neighborhood cells and perform a R(1) in all the t-base cells after each state transition;
10. bring the A,B,C,D-neighborhood cells in the all 1-state with an additional W(1) of the A-cells;

mode 2:

11. perform a R(1) in all the t-base cells;
12. sequence the fifteen state transitions according to path H3 in the a,b,c,d-neighborhood cells and perform a R(1) in all the T-base cells after each state transition

end.

APPENDIX B : Mealy machine, Hamiltonian and Eulerian paths

Some terms of electrical switching theory were used in section 3.6. They will be explained in this appendix.

B.1 Mealy Machine

Hayes drew in [30] a parallel between a memory and a sequential state machine by which the content of the memory cells can be viewed as the respective machine states. This leads for example for a two-cell memory M_2 (cell y_0 and y_1) to the following state diagram and state table [lit.b].

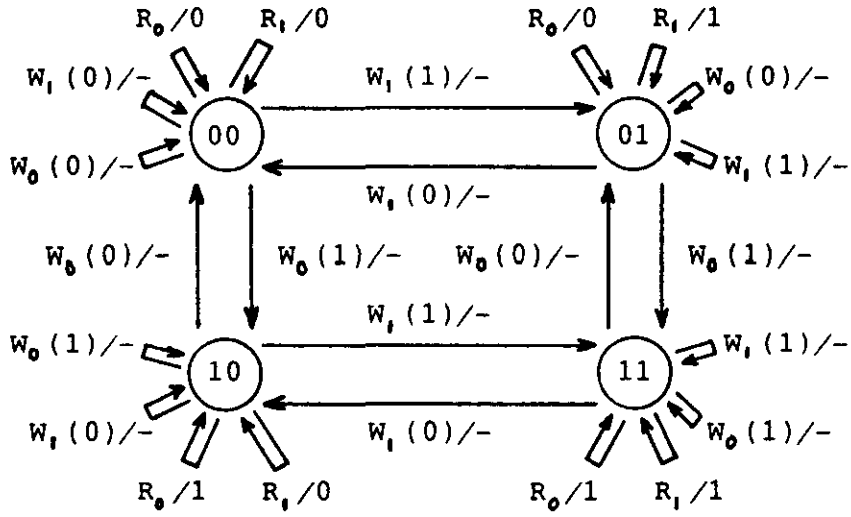


Fig.b1 State diagram for M_2. The abbreviation $R_0/1$ stands for a read of cell y_0, and the espected data has the logical 1-value, whereas $W_0(1)/-$ means that a write is performed in cell y_0 with the logical 1-value, the data output has an undefined level.

old value $y_0 y_1$	input operation					
	$W_0(1)$	$W_1(1)$	$W_0(0)$	$W_1(0)$	R_0	R_1
00	10,-	01,-	00,-	00,-	00,0	00,0
01	11,-	01,-	01,-	00,-	01,0	01,1
10	10,-	11,-	00,-	10,-	10,1	10,0
11	11,-	11,-	01,-	10,-	11,1	11,1

Table b1 State table for M_2. In the notation xx,-/0/1 both x's render the new state of $y_0 y_1$ and the symbol after the comma gives an indication of the expected data value on the output.

B.2 Hamiltonian Path

Hamiltonian Path is an optimization method by which all states in a sequential machine are sequenced with a minimal number of write operations. The binary values of two consequtive states may only differ in one bit position, i.e. with Hamming distance 1. If all 2^n states have to be traversed, then also 2^n transitions will suffice. The Gray codes are derived in this way, frequently used in computer and telecommunication applications.
Here we will show the properties of a Hamiltonian path by means of a Karnaugh map. For the 3 variables a,b and c, all 8 states can be viewed in the map of figure b2. Several paths can be constructed addressing each state only one time, Table b2 gives some examples. Hamiltonian paths for arbitrary N can be derived in a similar way.

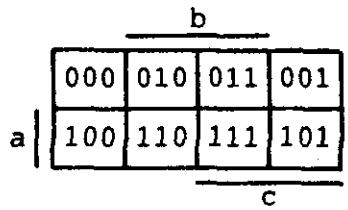


Fig.b2 Karnaugh map with the variables a,b,c.

000	000	000	111	111
010	100	100	101	110
011	110	101	001	100
001	010	111	011	101
101	011	110	010	001
111	111	010	000	000
110	101	011	100	010
100	001	001	110	011

Table b2 Some examples of Hamiltonian paths.

B.3 Eulerian Path

In difference with an optimal state sequence, the Eulerian path guarantees a minimal number of write actions to perform all transitions between the nodes in a state diagram. Figure b1 is then reduced to the subgraph of figure b3.

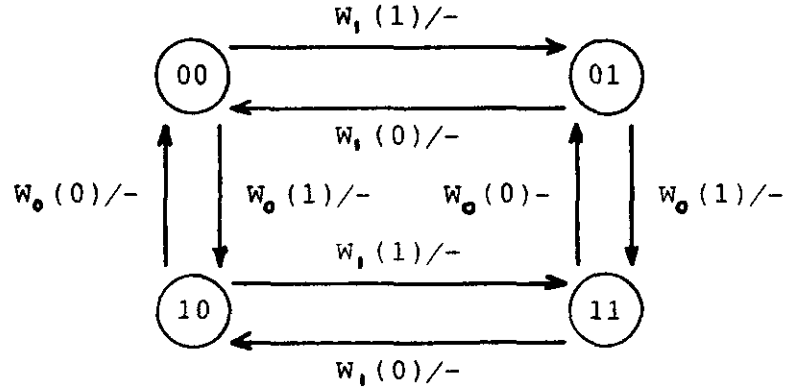
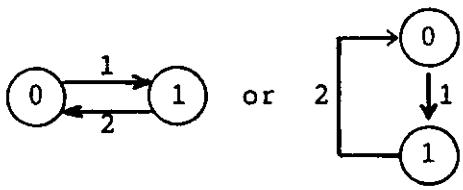


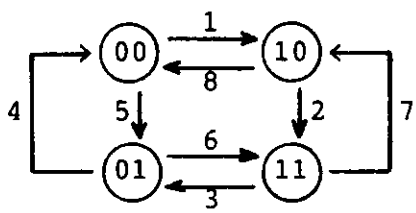
Fig.b3 Transition state diagram for M_2.

It is obvious that at most $\binom{n}{1}$ arrows leave a node in the diagram, whereby n stands for the number of bits representing the states. According the memory model of Hayes all 2^n states occur, hence the minimal number of writes is $n.2^n$ or n.N.

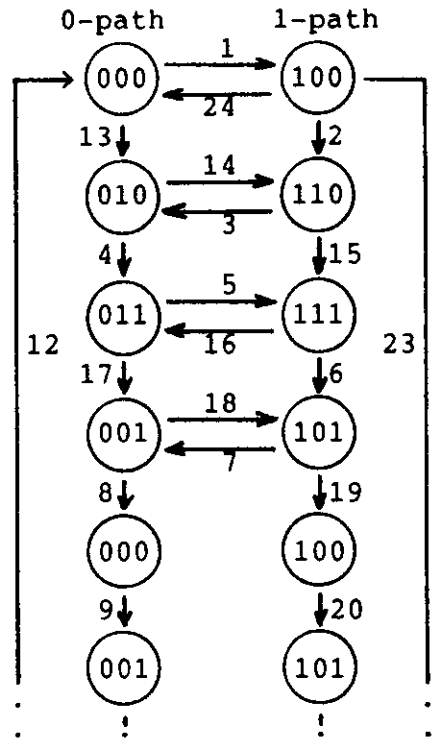
Suk and Reddy [42] explained in a simple way the generation of these Eulerian paths for arbitrary n out of paths for n-1, see figure b4.



Eulerian path for n=1



Eulerian path for n=2



Eulerian path for n=3

Figure b4
Eulerian paths for n=1,2,3. The path for n=2 is derived with the path for n=1, and the path for n=3 using the path for n=2.

The path for n-1 is dupliacated by adding a 0 and 1 in front, the 0-path and 1-path respectively. For each path 2^{n-1} crossover arcs are added. The number of transitions for the 0-path, the 1-path and the crossovers then comes to a total of:

$$(n-1).2^{n-1} + (n-1).2^{n-1} + 2.2^{n-1} = n.2^n \text{ arcs.}$$

It is easy to see that all possible transitions are traversed exactly once.

[lit.b] F.J. Hill, G.R. Peterson; Switching Theory and Logical Design. 3rd ed. , John Wiley and Sons, New York, USA, 1981, pp.300-305.

APPENDIX C : Timing diagrams for Philips' 16K SRAM

This appendix contains the timing diagrams which were used in both test programs.
Each set contains a read, a write and a dummy format. Dummy actions are passive read operations. They are sometimes necessarily to initialize the control lines or to adjust the address counters during the execution of an address pattern, see also appendix D. They do normally not affect the content of the memory.

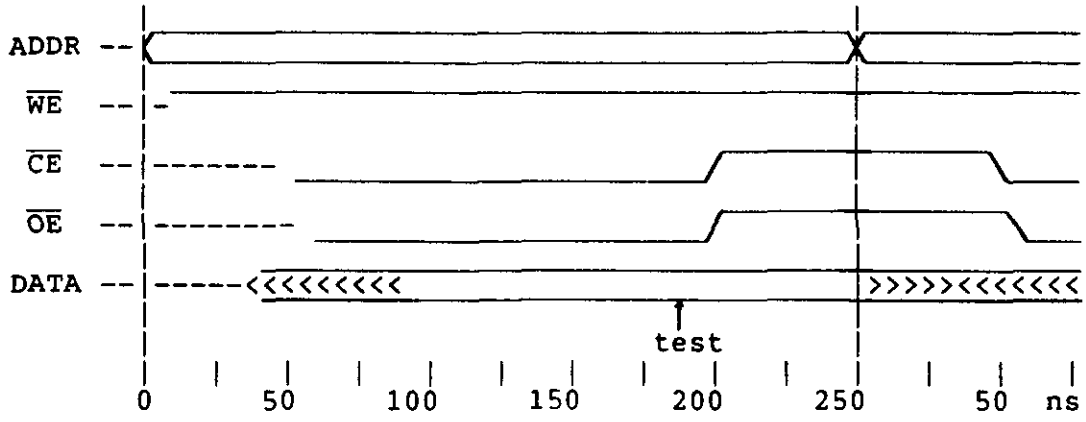
The functional diagram gives a relaxed timing of the memory to exclude most of the time-dependent faults. The control lines are executed according to the CE-latched mode. This ensures a synchronous operation by what the address access problems are excluded.

The CE-controlled mode gives a synchronous operation according to the specifications. Now the memory has been made sensitive to internal timing problems to detect some types of time-dependent faults.

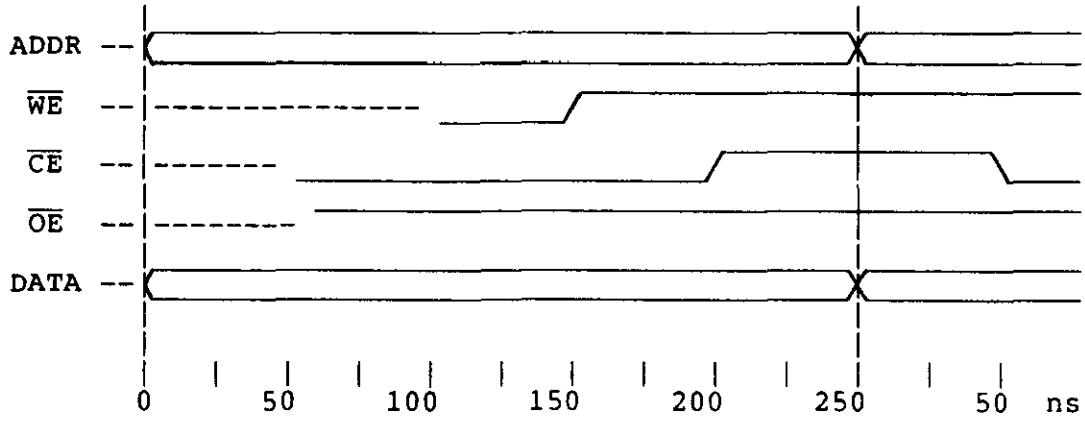
The address access mode differs only from the CE-controlled mode concerning the control of the CE-line. The operation of the memory is in this mode controlled by changes of the address lines, which results in an asynchronous timed device.

FUNCTIONAL TIMING DIAGRAMS:

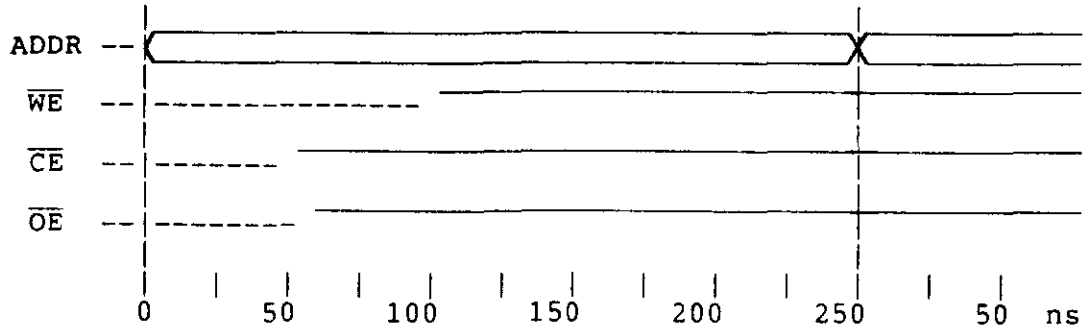
The read format:



The write format:

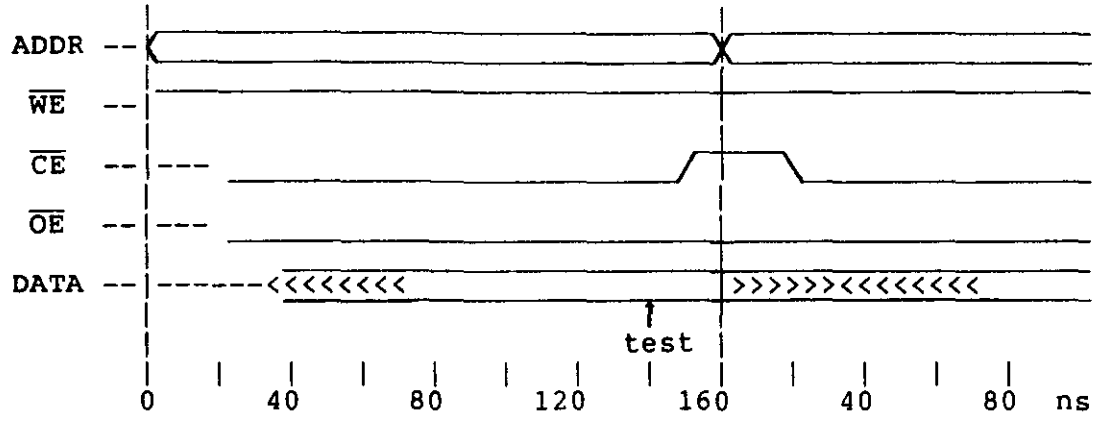


The dummy format:

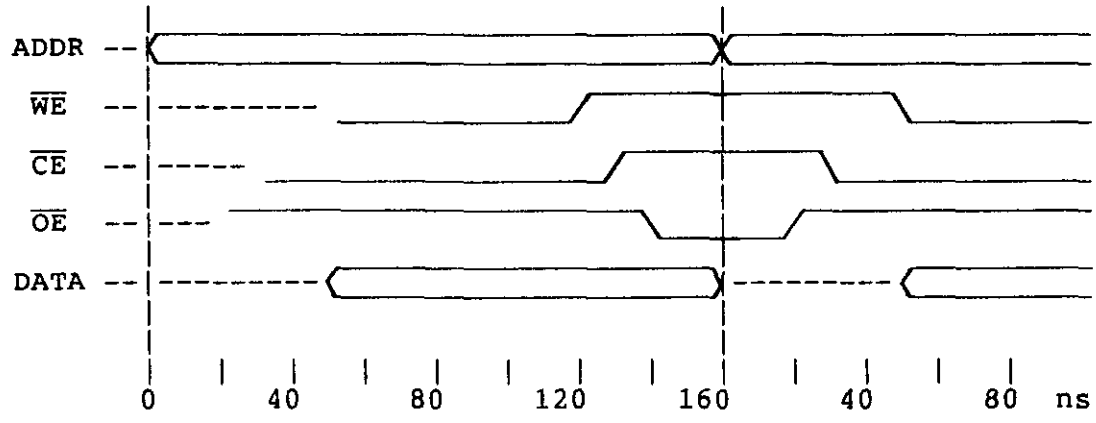


TIMING DIAGRAMS WITH $\overline{CE}$-CONTROL:

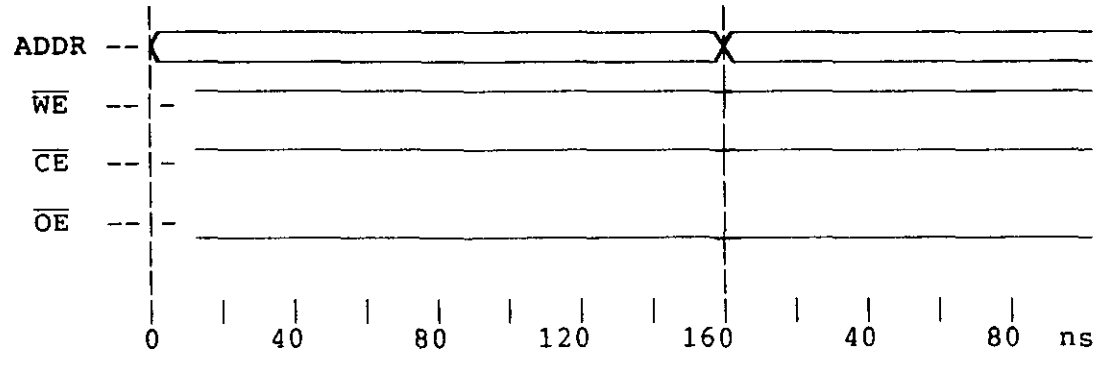
The read format:



The write format:

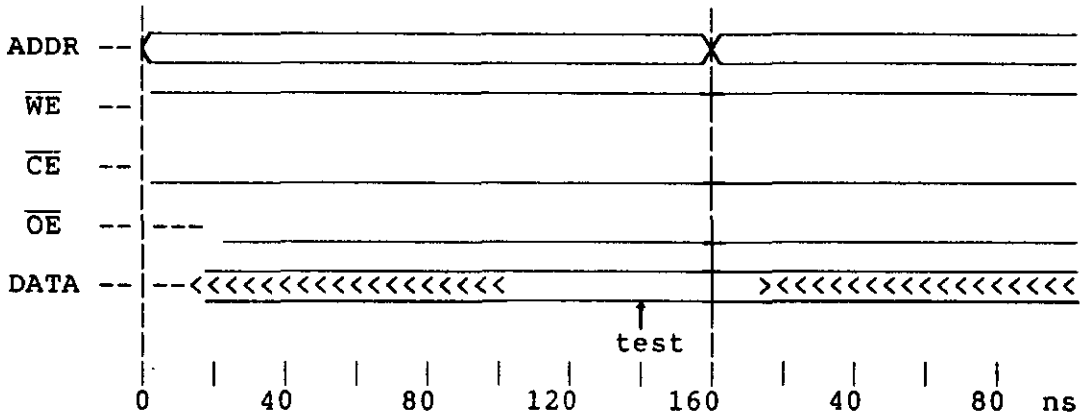


The dummy format:

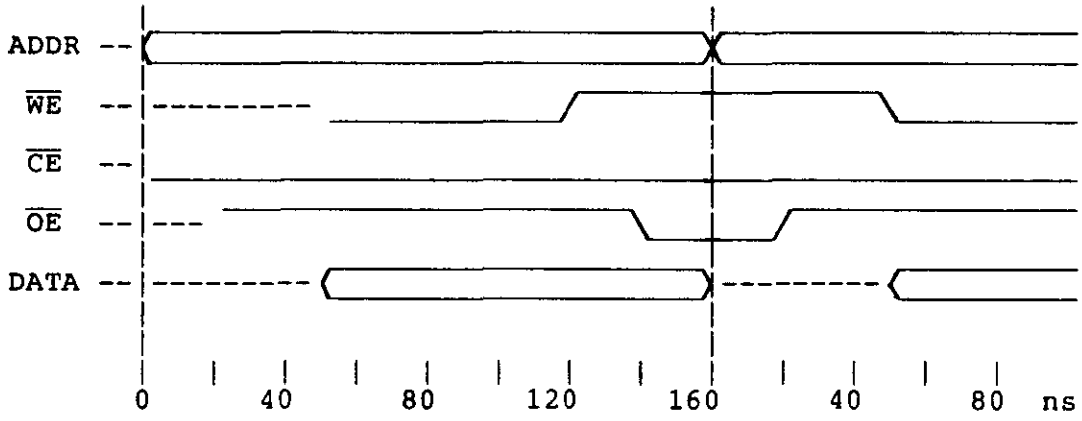


TIMING DIAGRAMS WITH ADDRESS ACCESS:

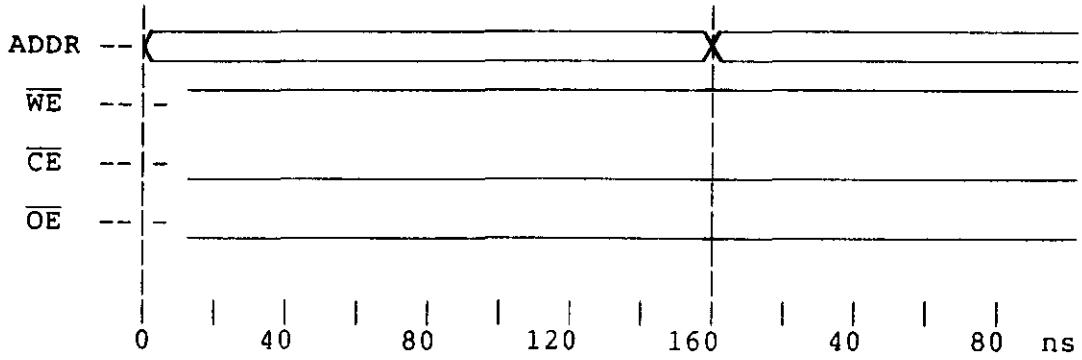
The read format:



The write format:



The dummy format:



APPENDIX D : Some implementations of patterns in PASCAL-t

Some pattern generator programs are given for the algorithms MSCAN, GALTCOL, Shifted Diagonal, Nair's MATS+ and Suk and Reddy's TEST B, as an illustration of section 4.2.5. Also, prints of the addressing sequences are taken up of Y-MSCAN and X-MSCAN, which may be used to verify the correctness of the pattern programs.

The MSCAN algorithms require 4N address accesses and prove that at least one cell exists in the whole memory array which can store both logical values. The decoder won't be checked adequately, so the strength of these procedures is questionable.
Y-MSCAN contains an addressing sequence that follows the word line, i.e.#the column decoder is switched quickly. The address generation of the X-MSCAN procedure follows the bit line and switches mostly the row decoder.

```
**************          // Ensure that both the main and the preset
Y-MSCAN, PCL 0          // counter are reset and the compare counter
**************          // is set to the maximum address value.
```

Add. Source	Counter Changes	Cond./Action	Branch	Data I/O
SOURCE XM YM			BRN .+1	DUMST
	initialisation	am=0		
SOURCE XM YM	INC X Y MOVE YLXM	UNTIL AM=AC	BRN .	MWRITE
	read sequence	am=0		background
SOURCE XM YM	INC X Y MOVE YLXM	UNTIL AM=AC	BRN .	MREAD
	initialisation	am=0		background
SOURCE XM YM	INC X Y MOVE YLXM	UNTIL AM=AC	BRN .	INVIN MWRITE
	read sequence	am=0		inv.backgr.
SOURCE XM YM	INC X Y MOVE YLXM	UNTIL AM=AC	BRN .	INVEX MREAD
		am=0		inv.backgr.
SOURCE XM YM			BRN .+1	DUMST
SOURCE XM YM			HALT .	DUMST
ENDPCL				

```
**************          // Ensure that both the main and the preset
X-MSCAN, PCL 0          // counter are reset and the compare counter
**************          // is set to the maximum address value.
```

Add. Source	Counter Changes	Cond./Action	Branch	Data I/O
SOURCE XM YM			BRN .+1	DUMST
	initialisation	am=0		
SOURCE XM YM	INC X Y MOVE XLYM	UNTIL AM=AC	BRN .	MWRITE
	read sequence	am=0		background
SOURCE XM YM	INC X Y MOVE XLYM	UNTIL AM=AC	BRN .	MREAD
	initialisation	am=0		background
SOURCE XM YM	INC X Y MOVE XLYM	UNTIL AM=AC	BRN .	INVIN MWRITE

```
            | read sequence     | am=0          |          | inv.backgr.
SOURCE XM YM  INC X Y MOVE XLYM  UNTIL AM=AC     BRN .      INVEX MREAD
            |                   | am=0          |          | inv.backgr.
SOURCE XM YM                                     BRN .+1    DUMST
SOURCE XM YM                                     HALT .     DUMST
ENDPCL
```

The pattern GALTCOL is a shortened version of GALPAT(column). During the execution of line 4-7 only jumps between nibbles with the same Y-address are performed in stead of all possible address combinations as GALPAT does. The complexity of the procedure is directly proportional to N**3/2.

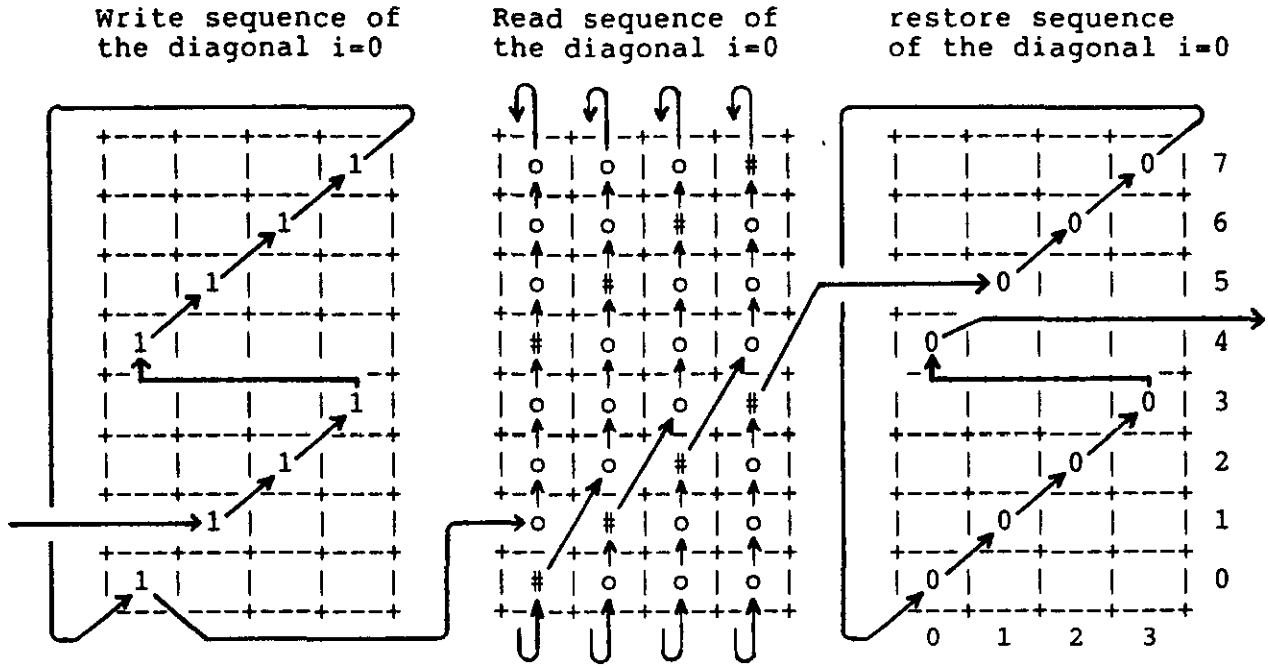
```
1. Write background data in nibble(m) for m=0,1,...,N-1
2. begin for m=0 until m=N-1
3.                do write inverted background data in
nibble(m);
4.                   for n=0 until "end of column" and m=/n
5.                       do read test nibble(m);
6.                          read nibble(n)
7.                       od;
8.                   read inverted background data in
nibble(m)
9.                   write background data in nibble(m)
10.               od
11.      end
```

```
**************          // Ensure that the main, the reference and the
GALTCOL, PCL 0          // preset counter are reset and the compare
**************          // counter is set to the maximum address value
                        // A normal and a complement data background
                        // are required to complete the test.
```

```
Add. Source | Counter Changes   | Cond./Action | Branch   | Data I/O
------------|------------------|--------------|----------|-----------
SOURCE XM YM                                     BRN .+1    DUMST
            | statement line 1  | am=ar=0      |          |
SOURCE XM YM  INC X Y MOVE XLYM  UNTIL AM=AC     BRN .      MWRITE
            | statement line 3  | am=ar=0      |          | background
TESTNIBBLE,
SOURCE XR YR  INC X MOVE XM                      BRN .+1    INVIN MWRITE
            | statements 5+6    | line 4       |          | test=inv.bg.
SOURCE XR YR  DEC X MOVE XR                      BRN .+1    INVEX MREAD
SOURCE XM YM  INC X MOVE XM XR   UNTIL XM=XR     BRN .-1    MREAD
            | statements 8+9    | line 2       |          | test=inv.bd
SOURCE XR YR                                     BRN .+1    INVEX MREAD
SOURCE XR YR  INC X Y MOVE       UNTIL AR=AC     BRN        MWRITE
                    XLYM XLYR                    TESTNIBB
            | statement line 11 | am=ar=0      |          | background
SOURCE XM YM                                     BRN .+1    DUMST
SOURCE XM YM                                     HALT .     DUMST
ENDPCL
```

DIAGON(column) stands for an implementation of the Shifted Diagonal pattern whereby a test diagonal is written with complemented background data. During the verification step the addresses are sequenced with a fast switched X-decoder.

The following capture elucidate for the first test diagonal the detailed addressing sequence in case the maximum X and Y values would be 7 and 3, respectively. For other maximum values the addressing sequence is principally the same.




```
****************        // Ensure that the main, the reference and the
DIAGONcol, PCL 0        // preset counter are reset and the compare
****************        // counter is set to the maximum address value
                        // A normal and a complement data background
                        // are required to complete the test. Above,
                        // the content of the loop counter should be
                        // equal to the maximum Y-address.
```

Add. Source	Counter Changes	Cond./Action	Branch	Data I/O
SOURCE XM YM			BRN .+1	DUMST
	initialisation	am=ar=0 lc=Ym		
SOURCE XM YM	INC X Y MOVE XLYM	UNTIL AM=AC	BRN .	MWRITE
		am=ar=0 lc=Ym		background
SOURCE XM YM	INC X Y MOVE XM YM YR		BRN .+1	DUMST
SOURCE XM YM	INC Y MOVE YR		BRN .+1	DUMST

```
                |                          |                  |                |
                |                          |am=[1,1] lc=Ym|                | background
                |                          |ar=[0,2]   i=0 |                |
DIAGON[i],
SOURCE XM YM INC X Y MOVE XM YM  UNTIL XM=XR     BRN .      INVIN MWRITE
                |                          |                  |                |
                |                          |am=[1-i+i*4,1]|                | background
                |                          |ar=[-i+i*4,2] |                | except
                |                          |lc=Ymax-i      |                | diagon[i]
                |                          |                  |                |
SOURCE XM YM DEC Y MOVE YM                        BRN .+1    DUMST
                |                          |                  |                |
                |                          |am=[1-i+i*4,0]|                | background
                |                          |ar=[-i+i*4,2] |                | except
                |                          |lc=Ymax-i      |                | diagon[i]
COLUMN[j],
SOURCE XM YM INC X Y MOVE XM YR  UNTIL YM=YR     BRN .      MREAD
SOURCE XM YM INC X Y MOVE XM YR  UNTIL XM=XR     BRN .-1    INVEX MREAD
                |                          |                  |                |
                |                          |am=[1+j-i+i*4,j]                | background
                |                          |ar=[j-i+i*4,2+j]                | except
                |                          |lc=Ymax-i      |                | diagon[i]
                |                          |                  |                |
SOURCE XM YM INC X Y MOVE XM YM  UNTIL YM=YC     BRN        DUMST
                          XR YR                   COLUMN[j]
                |                          | j=j+1           |                |
                |                          |am=[1+j-i+i*4,j]                | background
                |                          |ar=[j-i+i*4,2+j]                | except
                |                          |lc=Ymax-i      |                | diagon[i]
                |                          |                  |                |
                |                          | j=0             |                |
SOURCE XM YM INC Y MOVE YM                        BRN .+1    DUMST
                |                          |                  |                |
                |                          |am=[1-i+i*4,1]|                | background
                |                          |ar=[-i+i*4,2] |                | except
                |                          |lc=Ymax-i      |                | diagon[i]
                |                          |                  |                |
SOURCE XM YM INC X Y MOVE XM YM  UNTIL XM=XR     BRN .      MWRITE
                |                          |                  |                |
                |                          |am=[1-i+i*4,1]|                | background
                |                          |ar=[-i+i*4,2] |                |
                |                          |lc=Ymax-i      |                |
                |                          |                  |                |
SOURCE XM YM DEC X MOVE XM XR    UNTIL LCNR=0    BRN        DUMST
                                                  DIAGON[i]
                |                          | i=i+1           |                |
                |                          |am=[1-i+i*4,1]|                | background
                |                          |ar=[-i+i*4,2] |                |
                |                          |lc=Ymax-i      |                |
                |                          |                  |                |
                |                          |lc=0             |                |
SOURCE XM YM                                      BRN .+1    DUMST
SOURCE XM YM                                      HALT .     DUMST
ENDPCL
```

The MATS test procedures should have the same fault coverage properties as the MARCHING 0's/1's algorithm, but requires only 4N memory operations. If the logical behaviour of the decoder is unknown (wired-AND or wired-OR), the complexity increases to 5N. A detailed description of the pattern can be found in Appendix A.
The addressing sequence is equal to that of X-MSCAN.

```
***************          // Ensure that both the main and the preset
MATSplus, PCL 0          // counter are reset and the compare counter
***************          // is set to the maximum address value.
```

Add. Source	Counter Changes	Cond./Action	Branch	Data I/O
SOURCE XM YM			BRN .+1	DUMST
	initialisation	am=0		
SOURCE XM YM	INC X Y MOVE XLYM	UNTIL AM=AC	BRN .	MWRITE
	sequence 1	am=0		background
SOURCE XM YM			BRN .+1	MREAD
SOURCE XM YM	INC X Y MOVE XLYM	UNTIL AM=AC	BRN .-1	INVIN MWRITE
	sequence 2	am=0		inv.backgr.
SOURCE XM YM			BRN .+1	INVEX MREAD
SOURCE XM YM	INC X Y MOVE XLYM	UNTIL AM=AC	BRN .-1	MWRITE
		am=0		background
SOURCE XM YM			BRN .+1	DUMST
SOURCE XM YM			HALT .	DUMST
ENDPCL				

The GMATS-plus pattern differs only from the MATS-plus pattern concerning the addressing sequence. The main counter accesses nibbles with a fast changing of the Y-addresses, whereas the reference counter causes fast switching of the X-addresses. The resulting address sequence is GALPAT like.

```
****************        // Ensure that the main, reference and the
GMATSplus, PCL 0        // preset counter are reset, whereas the
****************        // compare counter is set to the maximum value

Add. Source  | Counter Changes    | Cond./Action  | Branch   | Data I/O
-------------|--------------------|---------------|----------|-----------
SOURCE XM YM                                       BRN .+1    DUMST
             | initialisation     | am=ar=0       |          |
             |                    |               |          |
SOURCE XM YM  INC X Y MOVE YM XR IF YM=YC           BRN .+4    MWRITE
SOURCE XR YR  INC X Y MOVE YM XR UNTIL YM=YC        BRN .-1    MWRITE
SOURCE XR YR  INC X Y MOVE XM YM                   BRN .-2    DUMST
SOURCE XM YM  INC X Y MOVE YM YR                   BRN .-2    MWRITE
SOURCE XR YR  INC X Y MOVE XM XR IF XR=XC          BRN .-1    MWRITE
SOURCE XR YR                      UNTIL XR=XC      BRN .-5    DUMST
SOURCE XR YR  INC X Y MOVE XLYR   UNTIL AR=AC      BRN .-6    DUMST
             |                    |               |          |
             | sequence 1         | am=ar=0       |          | background
             |                    |               |          |
SOURCE XM YM                                       BRN .+1    MREAD
SOURCE XM YM  INC X Y MOVE YM XR IF YM=YC           BRN .+6    INVIN MWRITE
SOURCE XR YR                                       BRN .+1    MREAD
SOURCE XR YR  INC X Y MOVE YM XR UNTIL YM=YC        BRN .-3    INVIN MWRITE
SOURCE XR YR  INC X Y MOVE XM YM                   BRN .-4    DUMST
SOURCE XM YM                                       BRN .+1    MREAD
SOURCE XM YM  INC X Y MOVE YM YR                   BRN .-4    INVIN MWRITE
SOURCE XR YR                                       BRN .+1    MREAD
SOURCE XR YR  INC X Y MOVE XM XR IF XR=XC          BRN .-3    INVIN MWRITE
SOURCE XR YR                      UNTIL XR=XC      BRN .-11   DUMST
SOURCE XR YR  INC X Y MOVE XLYR   UNTIL AR=AC      BRN .-12   DUMST
             |                    |               |          |
             | sequence 2         | am=ar=0       |          | inv.backgr.
             |                    |               |          |
SOURCE XM YM                                       BRN .+1    INVEX MREAD
SOURCE XM YM  INC X Y MOVE YM XR IF YM=YC           BRN .+6    MWRITE
SOURCE XR YR                                       BRN .+1    INVEX MREAD
SOURCE XR YR  INC X Y MOVE YM XR UNTIL YM=YC        BRN .-3    MWRITE
SOURCE XR YR  INC X Y MOVE XM YM                   BRN .-4    DUMST
SOURCE XM YM                                       BRN .+1    INVEX MREAD
SOURCE XM YM  INC X Y MOVE YM YR                   BRN .-4    MWRITE
SOURCE XR YR                                       BRN .+1    INVEX MREAD
SOURCE XR YR  INC X Y MOVE XM XR IF XR=XC          BRN .-3    MWRITE
SOURCE XR YR                      UNTIL XR=XC      BRN .-11   DUMST
SOURCE XR YR  INC X Y MOVE XLYR   UNTIL AR=AC      BRN .-12   DUMST
             |                    |               |          |
             |                    | am=ar=0       |          | background
             |                    |               |          |
SOURCE XM YM                                       BRN .+1    DUMST
SOURCE XM YM                                       HALT .     DUMST
ENDPCL
```

Suk and Reddy's TEST B takes 17N memory actions including an initialization step. The pattern is programmed with the same addressing sequence as X-MSCAN. A detailed description is given in Appendix A.

```
*************          // Ensure that both the main and the preset
SUKRED, PCL 0          // counter are reset and the compare counter
*************          // is set to the maximum address value.

Add. Source  | Counter Changes   | Cond./Action | Branch   | Data I/O
-------------|-------------------|--------------|----------|-----------
SOURCE XM YM                                      BRN .+1    DUMST
             | initialisation    | am=ar=0      |          |
             |                   |              |          |
SOURCE XM YM | INC X Y MOVE XLYM | UNTIL AM=AC  | BRN .    | MWRITE
             |                   |              |          |
             | sequence 1        | am=0         |          | background
SOURCE XM YM                                      BRN .+1    MREAD
SOURCE XM YM                                      BRN .+1    INVIN MWRITE
SOURCE XM YM                                      BRN .+1    INVEX MREAD
SOURCE XM YM                                      BRN .+1    MWRITE
SOURCE XM YM                                      BRN .+1    MREAD
SOURCE XM YM | INC X Y MOVE XLYM | UNTIL AM=AC  | BRN .-5   | INVIN MWRITE
             |                   |              |          |
             | sequence 2        | am=0         |          | inv.backgr.
SOURCE XM YM                                      BRN .+1    INVEX MREAD
SOURCE XM YM                                      BRN .+1    MWRITE
SOURCE XM YM | INC X Y MOVE XLYM | UNTIL AM=AC  | BRN .-2   | INVIN MWRITE
             |                   |              |          |
             | sequence 3        | am=0         |          | inv.backgr.
SOURCE XI YI                                      BRN .+1    INVEX MREAD
SOURCE XI YI                                      BRN .+1    MWRITE
SOURCE XI YI                                      BRN .+1    INVIN MWRITE
SOURCE XI YI | INC X Y MOVE XLYM | UNTIL AM=AC  | BRN .-3   | MWRITE
             |                   |              |          |
             | sequence 4        | am=0         |          | background
SOURCE XI YI                                      BRN .+1    MREAD
SOURCE XI YI                                      BRN .+1    INVIN MWRITE
SOURCE XI YI | INC X Y MOVE XLYM | UNTIL AM=AC  | BRN .-2   | MWRITE
             |                   |              |          |
SOURCE XM YM                                      BRN .+1    DUMST
SOURCE XM YM                                      HALT .     DUMST
ENDPCL
```

Printed addressing sequences of the first two statements of the pattern Y-MSCAN with Xmax=7 and Ymax=1. A dummy action can be recognized if CSET ESET FSET are "15 15 13", whereas in case of a write operation the values are equal to "1 1 1"

pcl	Xm	Ym	Xr	Yr	Xi	Yi	Source		Cset	Eset	Fset	Lcnr	In	Ex
0	0	0	0	0	7	1	Xm	Ym	15	15	13	14	0	0
1	0	0	0	0	7	1	Xm	Ym	1	1	1	14	0	0
1	0	1	0	0	7	0	Xm	Ym	1	1	1	14	0	0
1	1	0	0	0	6	1	Xm	Ym	1	1	1	14	0	0
1	1	1	0	0	6	0	Xm	Ym	1	1	1	14	0	0
1	2	0	0	0	5	1	Xm	Ym	1	1	1	14	0	0
1	2	1	0	0	5	0	Xm	Ym	1	1	1	14	0	0
1	3	0	0	0	4	1	Xm	Ym	1	1	1	14	0	0
1	3	1	0	0	4	0	Xm	Ym	1	1	1	14	0	0
1	4	0	0	0	3	1	Xm	Ym	1	1	1	14	0	0
1	4	1	0	0	3	0	Xm	Ym	1	1	1	14	0	0
1	5	0	0	0	2	1	Xm	Ym	1	1	1	14	0	0
1	5	1	0	0	2	0	Xm	Ym	1	1	1	14	0	0
1	6	0	0	0	1	1	Xm	Ym	1	1	1	14	0	0
1	6	1	0	0	1	0	Xm	Ym	1	1	1	14	0	0
1	7	0	0	0	0	1	Xm	Ym	1	1	1	14	0	0
1	7	1	0	0	0	0	Xm	Ym	1	1	1	14	0	0
2	0	0	0	0	7	1	Xm	Ym	0	0	0	14	0	0
2	0	1	0	0	7	0	Xm	Ym	0	0	0	14	0	0
2	1	.	.	.	.	.	.	.	.	.	.	.	.	.
.														

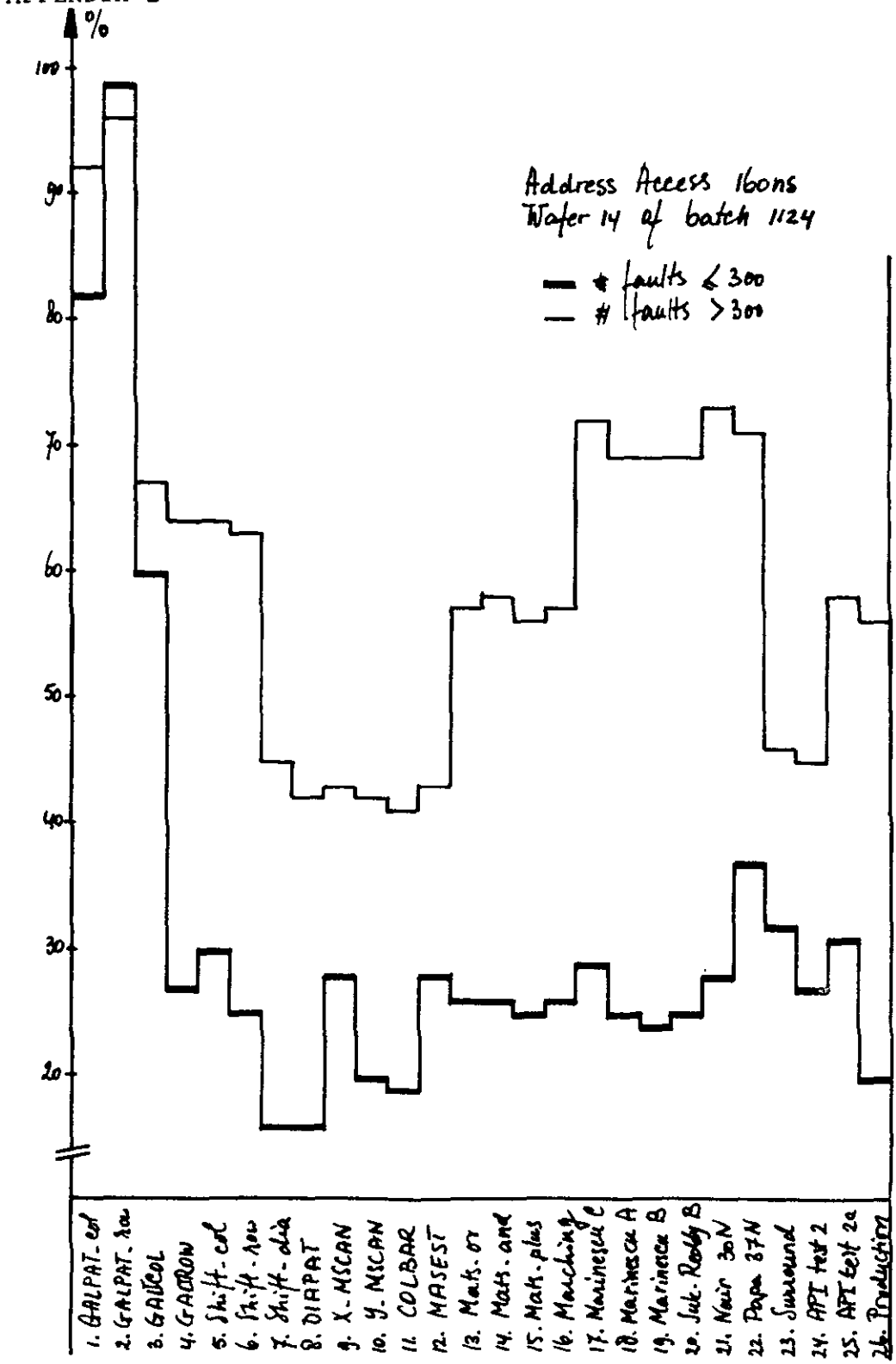
A similar result for the pattern X-MSCAN.

pcl	Xm	Ym	Xr	Yr	Xi	Yi	Source		Cset	Eset	Fset	Lcnr	In	Ex
0	0	0	0	0	7	1	Xm	Ym	15	15	13	14	0	0
1	0	0	0	0	7	1	Xm	Ym	1	1	1	14	0	0
1	1	0	0	0	6	1	Xm	Ym	1	1	1	14	0	0
1	2	0	0	0	5	1	Xm	Ym	1	1	1	14	0	0
1	3	0	0	0	4	1	Xm	Ym	1	1	1	14	0	0
1	4	0	0	0	3	1	Xm	Ym	1	1	1	14	0	0
1	5	0	0	0	2	1	Xm	Ym	1	1	1	14	0	0
1	6	0	0	0	1	1	Xm	Ym	1	1	1	14	0	0
1	7	0	0	0	0	1	Xm	Ym	1	1	1	14	0	0
1	0	1	0	0	7	0	Xm	Ym	1	1	1	14	0	0
1	1	1	0	0	6	0	Xm	Ym	1	1	1	14	0	0
1	2	1	0	0	5	0	Xm	Ym	1	1	1	14	0	0
1	3	1	0	0	4	0	Xm	Ym	1	1	1	14	0	0
1	4	1	0	0	3	0	Xm	Ym	1	1	1	14	0	0
1	5	1	0	0	2	0	Xm	Ym	1	1	1	14	0	0
1	6	1	0	0	1	0	Xm	Ym	1	1	1	14	0	0
1	7	1	0	0	0	0	Xm	Ym	1	1	1	14	0	0
2	0	0	0	0	7	1	Xm	Ym	0	0	0	14	0	0
2	1	0	0	0	6	1	Xm	Ym	0	0	0	14	0	0
2	2	.	.	.	.	.	.	.	.	.	.	.	.	.
.														

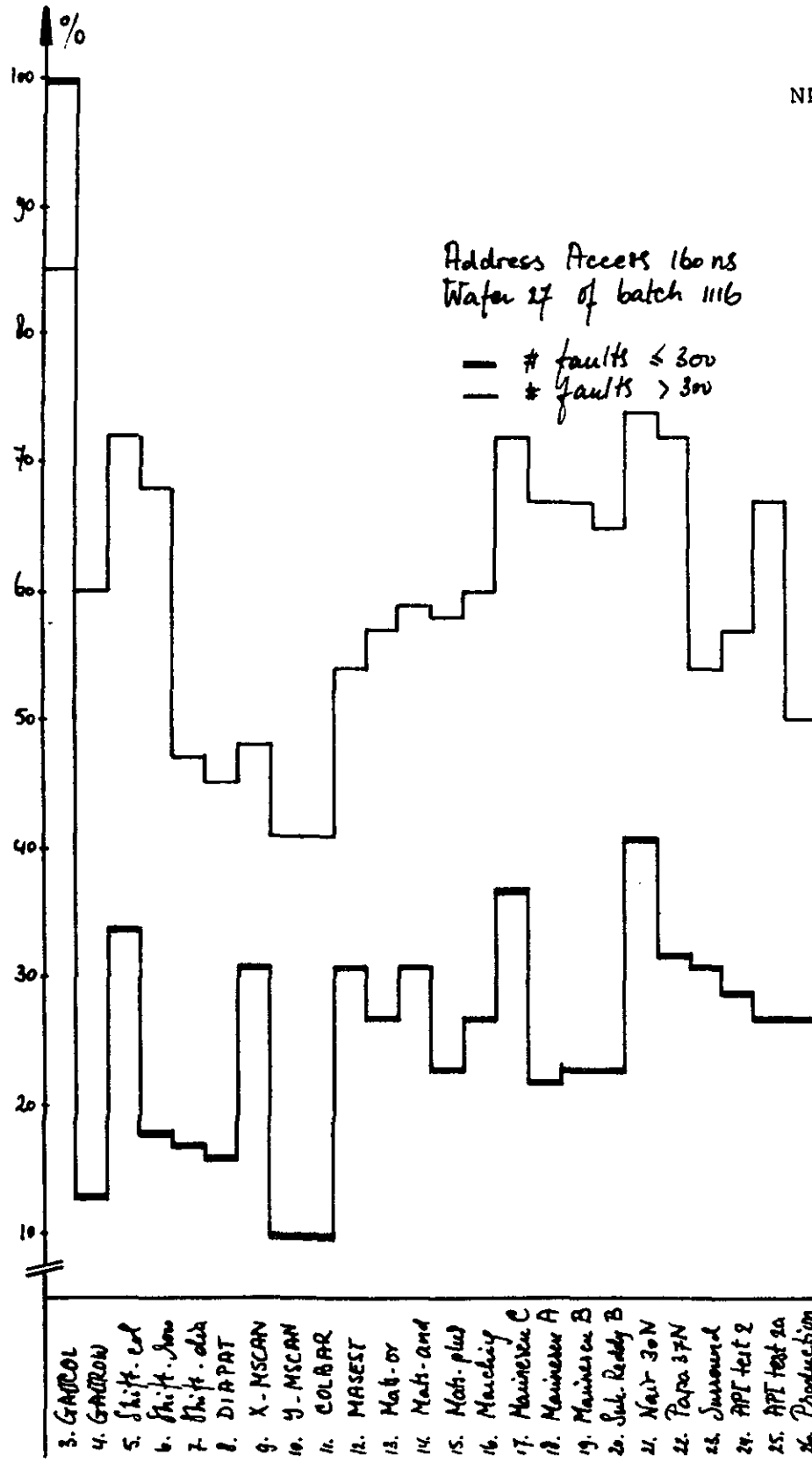
APPENDIX E : Graphic results of memory test program 1

This appendix contains a number of plots based on the results of test program 1, see section 5.3. The percentages don't represent the quality of a wafer, since only those memories have been selected for which the patterns displayed different fault numbers.

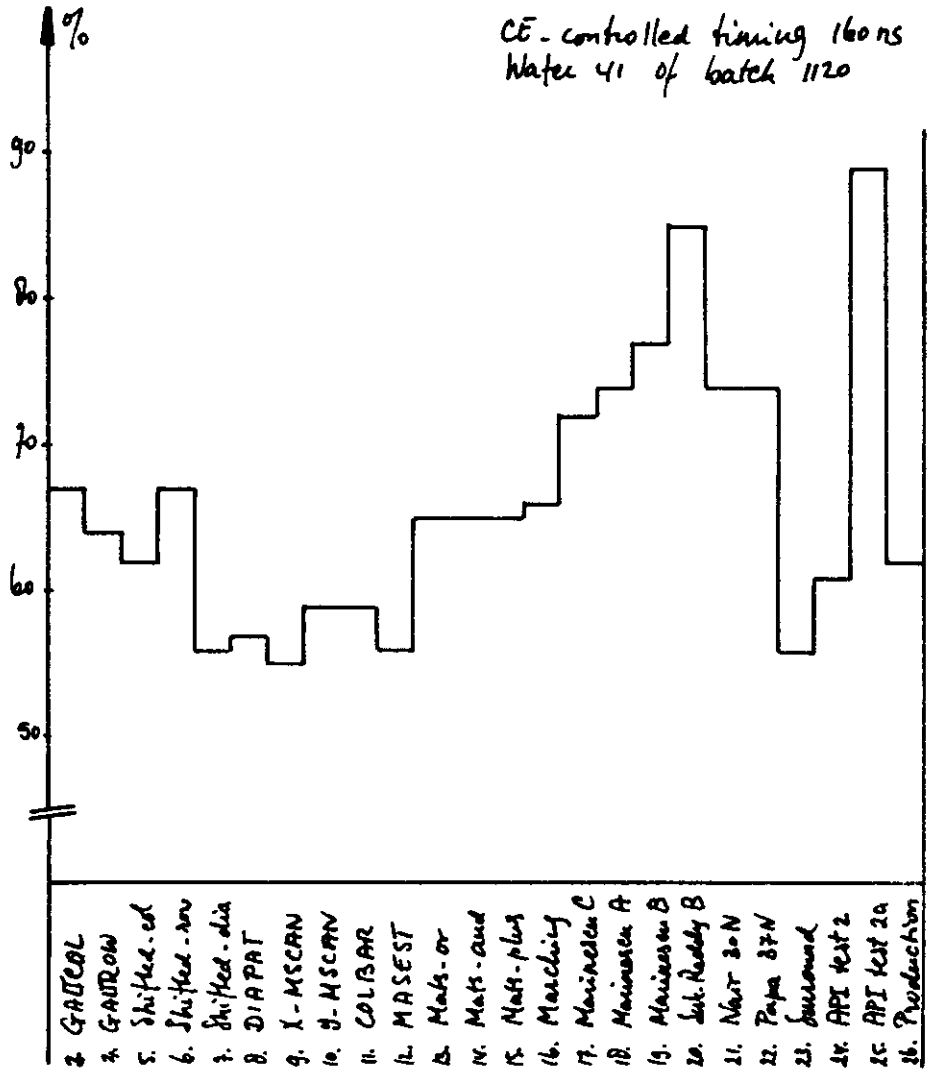
APPENDIX E



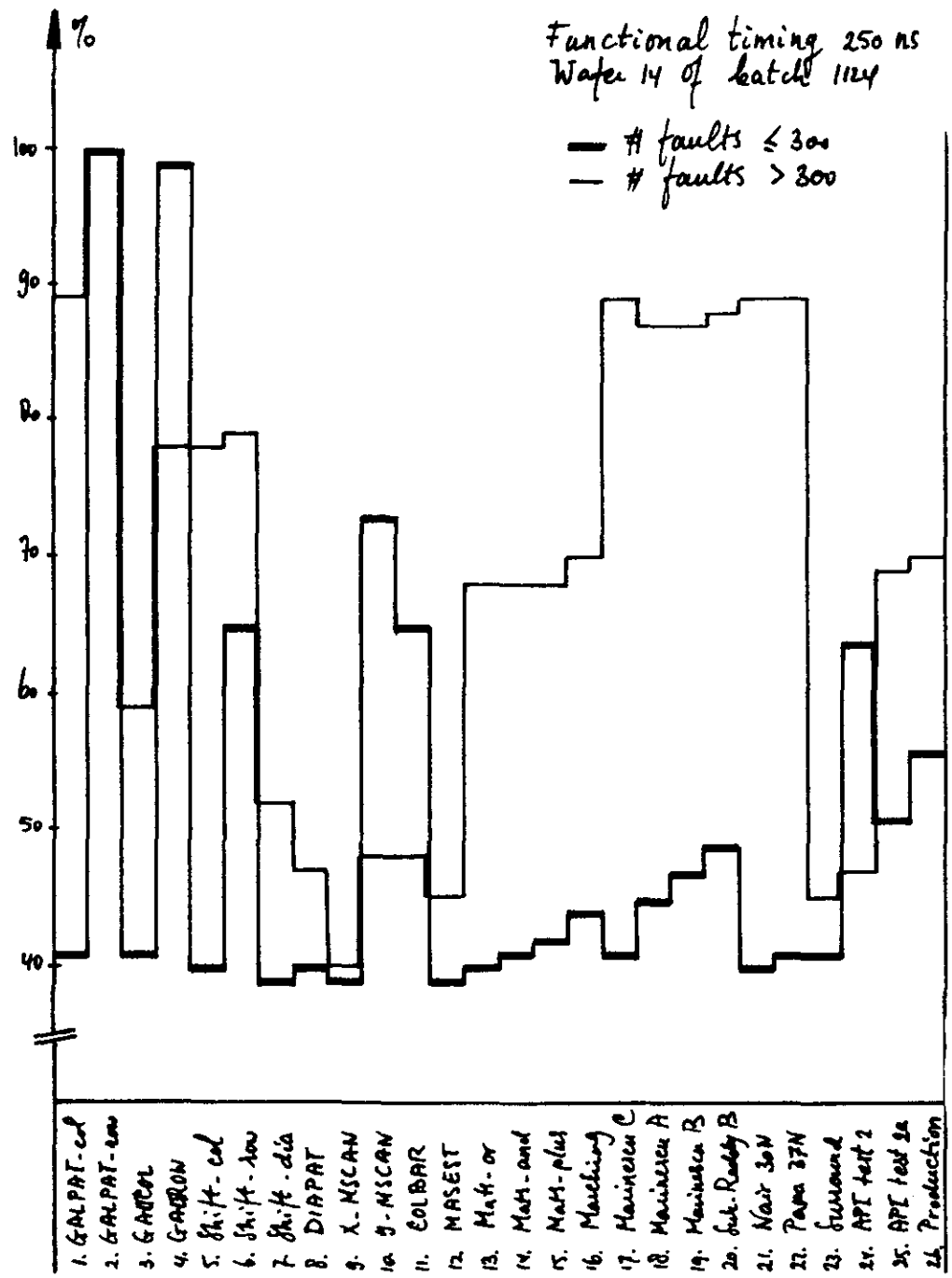
Plot 1 Average pattern set performance distribution with the address access timing at wafer 14 of batch 1124.



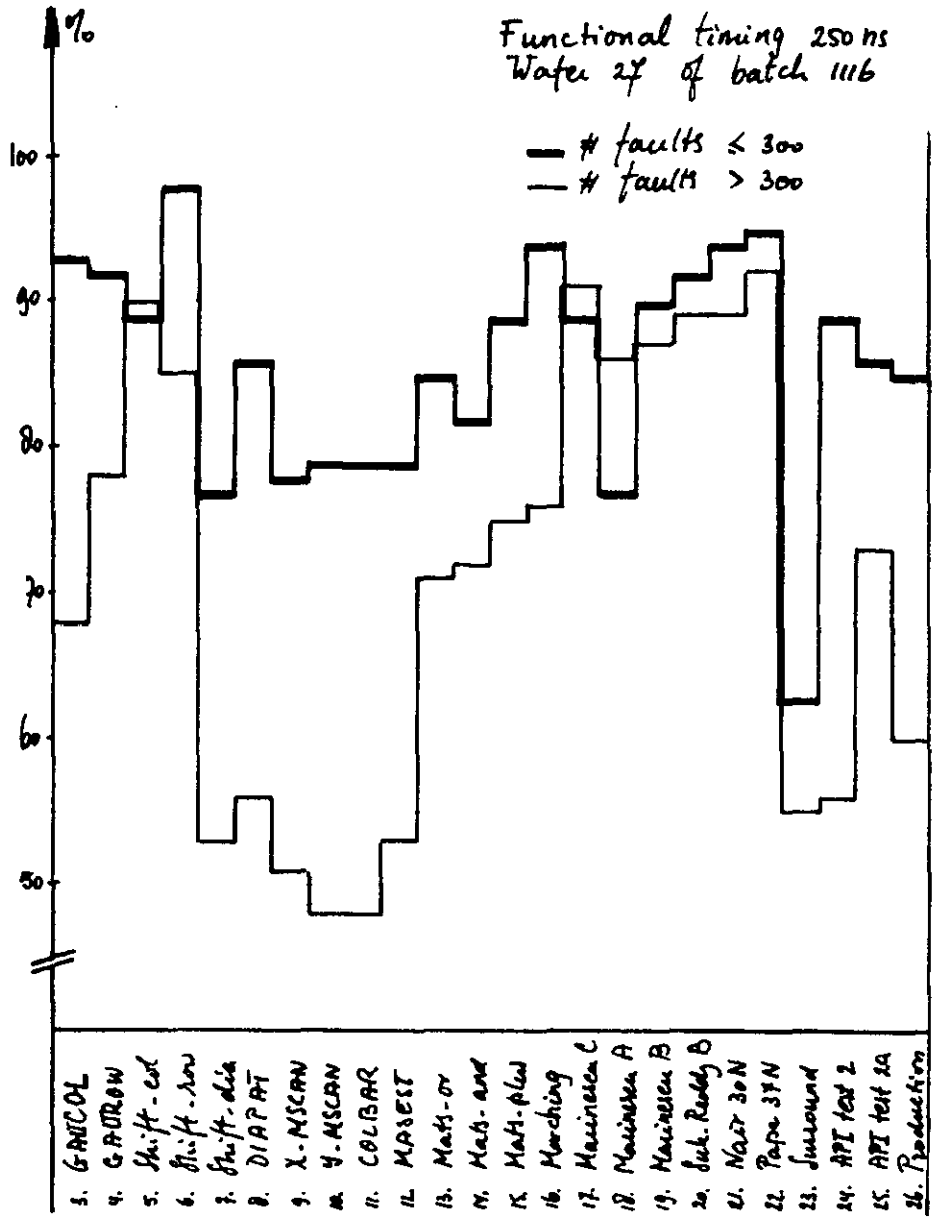
Plot 2 Average pattern set performance distribution without the GALPATs executed with the address access timing diagram at wafer 27 of batch 1116.



Plot 3 Average pattern set performance distribution with the CE-controlled timing at wafer 41 of batch 1120.



Plot 4 Average pattern set performance distribution with the functional timing diagram at wafer 14 of batch 1124.



Plot 5 Average pattern set performance distribution without the GALPATs executed with the functional timing diagram at wafer 27 of batch 1116.

APPENDIX F : Adressing orders and graphic results of memory test program 2

The MATSplus pattern has been used to examine the possible impact of eight addressing orders on the fault coverage. The results are discussed in section 5.4.
First, figure 1 shows the operation sequence of the MATSplus algorithm. Secondly, the standard nibble assignment order has been made visible in figure 2, based on a column following address generation. Thirdly, figure 3 gives outlines of the remaining seven address variations. At last, the appendix contains a number of plots representing the percentual distributions obtained with test program 2 on wafer 7 of batch 3174.

addr.	step 1	step 2	step 3
0	W(0)	R(0)W(1)	R(1)W(0)
1	W(0)	R(0)W(1)	R(1)W(0)
:	:	:	:
N-2	W(0)	R(0)W(1)	R(1)W(0)
N-1	W(0)	R(0)W(1)	R(1)W(0)

Fig.1 Operation sequence of the MATSplus algorithm.

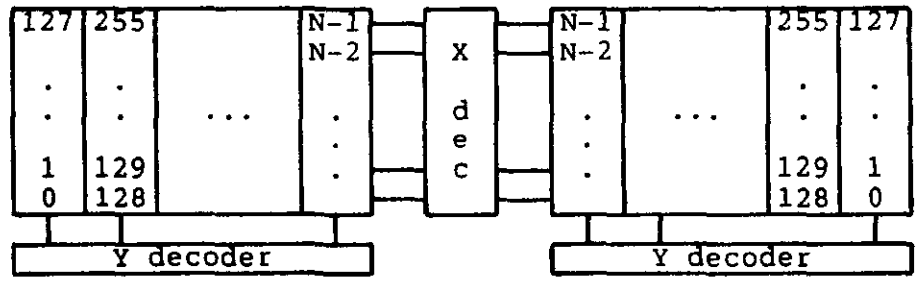
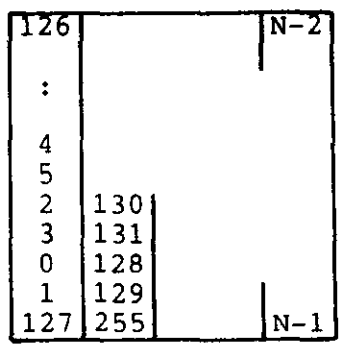
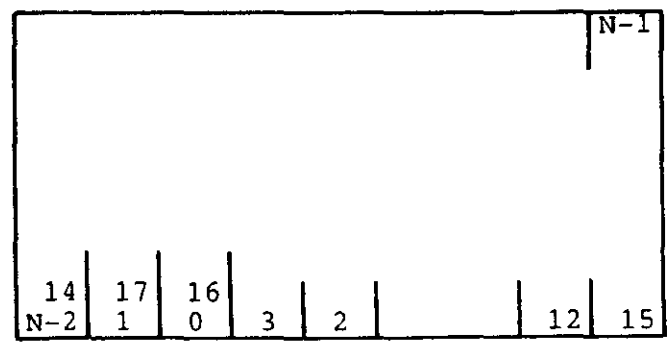


Fig.2 The usual addressing manner results in the successive access of the nibbles along the same bit line (column)

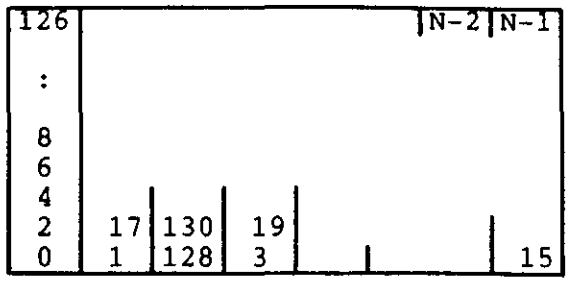
A-MATSplus:



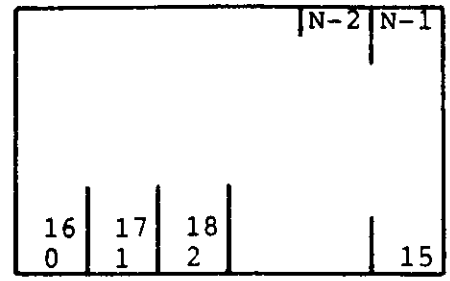
B-MATSplus:



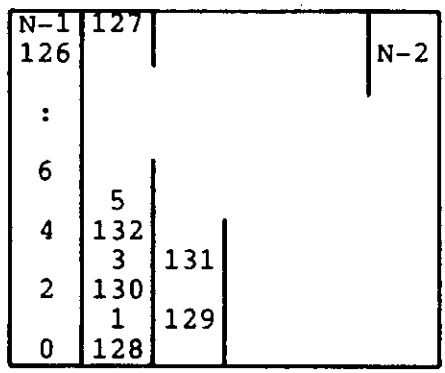
C-MATSplus:



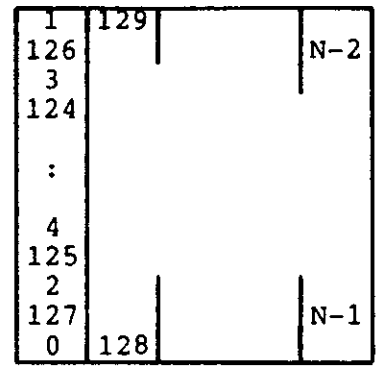
D-MATSplus:



E-MATSplus:



F-MATSplus:



G-MATSplus:

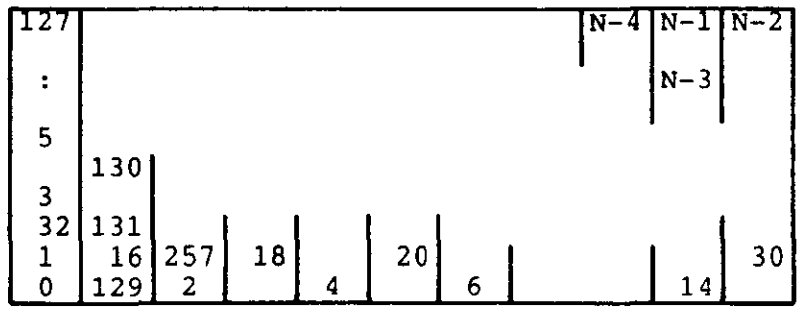
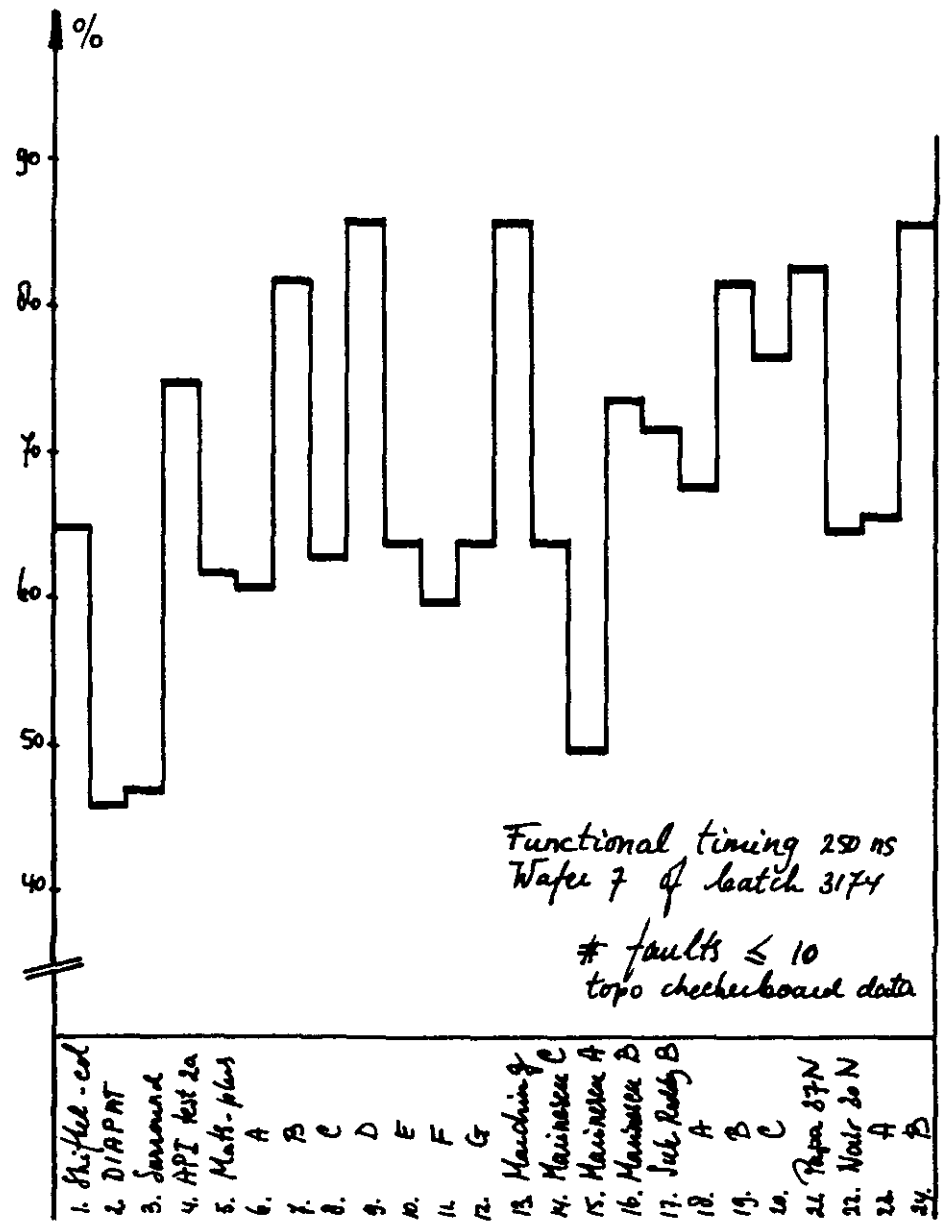
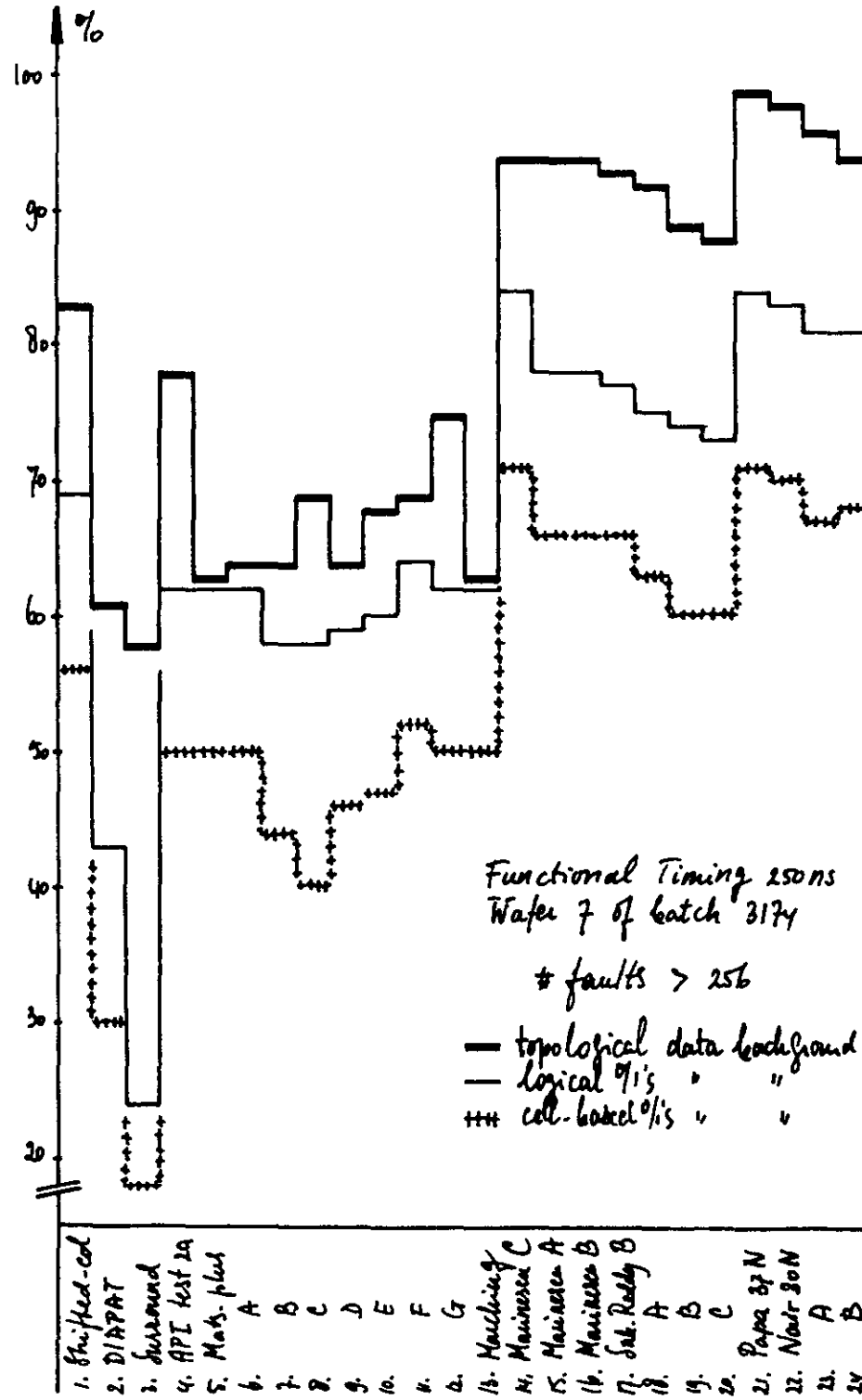


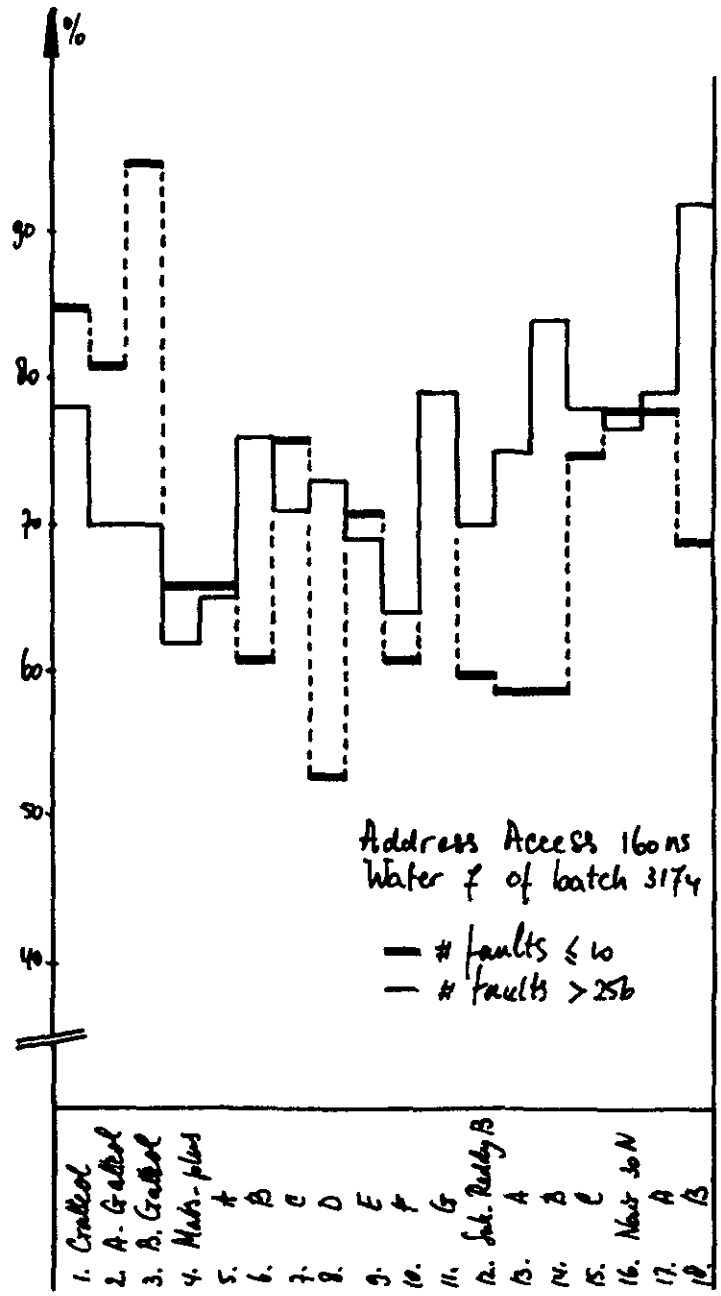
Fig.3 The remaining address variations exists of (a) address jumps within the columns, (b) address jumps within the rows, (c) the even numbers follow the columns whereas the odd numbers sequence the rows, (d) a row following addressing order, (e) address jumps between neighbouring columns, (f) alternating accesses between nibbles with a complement X-address and (g) the address method is similar to the C-MATSplus version but enables jumps between nibbles in the same column.



Plot 1 Average pattern set performance distribution with the functional timing diagram at wafer 7 of batch 3174.



Plot 2 Average pattern set performance distributions with the functional timing at wafer 14 of batch 1124.



Plot 3 Average pattern set performance distributions with the address access timing at wafer 7 of batch 3174.

ISSN 0167-9708
Coden: TEUEDE

(138) Nicola, V.F.
A SINGLE SERVER QUEUE WITH MIXED TYPES OF INTERRUPTIONS: Application to the modelling of checkpointing and recovery in a transactional system.
EUT Report 83-E-138. 1983. ISBN 90-6144-138-2

(139) Arts, J.G.A. and W.F.H. Merck
TWO-DIMENSIONAL MHD BOUNDARY LAYERS IN ARGON-CESIUM PLASMAS.
EUT Report 83-E-139. 1983. ISBN 90-6144-139-0

(140) Willems, F.M.J.
COMPUTATION OF THE WYNER-ZIV RATE-DISTORTION FUNCTION.
EUT Report 83-E-140. 1983. ISBN 90-6144-140-4

(141) Heuvel, W.M.C. van den and J.E. Daalder, M.J.M. Boone, L.A.H. Wilmes
INTERRUPTION OF A DRY-TYPE TRANSFORMER IN NO-LOAD BY A VACUUM CIRCUIT-BREAKER.
EUT Report 83-E-141. 1983. ISBN 90-6144-141-2

(142) Fronczak, J.
DATA COMMUNICATIONS IN THE MOBILE RADIO CHANNEL.
EUT Report 83-E-142. 1983. ISBN 90-6144-142-0

(143) Stevens, M.P.J. en M.P.M. van Loon
EEN MULTIFUNCTIONELE I/O-BOUWSTEEN.
EUT Report 84-E-143. 1984. ISBN 90-6144-143-9

(144) Dijk, J. and A.P. Verlijsdonk, J.C. Arnbak
DIGITAL TRANSMISSION EXPERIMENTS WITH THE ORBITAL TEST SATELLITE.
EUT Report 84-E-144. 1984. ISBN 90-6144-144-7

(145) Weert, M.J.M. van
MINIMALISATIE VAN PROGRAMMABLE LOGIC ARRAYS.
EUT Report 84-E-145. 1984. ISBN 90-6144-145-5

(146) Jochems, J.C. en P.M.C.M. van den Eijnden
TOESTAND-TOEWIJZING IN SEQUENTIËLE CIRCUITS.
EUT Report 85-E-146. 1985. ISBN 90-6144-146-3

(147) Rozendaal, L.T. en M.P.J. Stevens, P.M.C.M. van den Eijnden
DE REALISATIE VAN EEN MULTIFUNCTIONELE I/O-CONTROLLER MET BEHULP VAN EEN GATE-ARRAY.
EUT Report 85-E-147. 1985. ISBN 90-6144-147-1

(148) Eijnden, P.M.C.M. van den
A COURSE ON FIELD PROGRAMMABLE LOGIC.
EUT Report 85-E-148. 1985. ISBN 90-6144-148-X

(149) Beeckman, P.A.
MILLIMETER-WAVE ANTENNA MEASUREMENTS WITH THE HP8510 NETWORK ANALYZER.
EUT Report 85-E-149. 1985. ISBN 90-6144-149-8

(150) Meer, A.C.P. van
EXAMENRESULTATEN IN CONTEXT MBA.
EUT Report 85-E-150. 1985. ISBN 90-6144-150-1

(151) Ramakrishnan, S. and W.M.C. van den Heuvel
SHORT-CIRCUIT CURRENT INTERRUPTION IN A LOW-VOLTAGE FUSE WITH ABLATING WALLS.
EUT Report 85-E-151. 1985. ISBN 90-6144-151-X

(152) Stefanov, B. and L. Zarkova, A. Veefkind
DEVIATION FROM LOCAL THERMODYNAMIC EQUILIBRIUM IN A CESIUM-SEEDED ARGON PLASMA.
EUT Report 85-E-152. 1985. ISBN 90-6144-152-8

(153) Hof, P.M.J. Van den and P.H.M. Janssen
SOME ASYMPTOTIC PROPERTIES OF MULTIVARIABLE MODELS IDENTIFIED BY EQUATION ERROR TECHNIQUES.
EUT Report 85-E-153. 1985. ISBN 90-6144-153-6

(154) Geerlings, J.H.T.
LIMIT CYCLES IN DIGITAL FILTERS: A bibliography 1975-1984.
EUT Report 85-E-154. 1985. ISBN 90-6144-154-4

(155) Groot, J.F.G. de
THE INFLUENCE OF A HIGH-INDEX MICRO-LENS IN A LASER-TAPER COUPLING.
EUT Report 85-E-155. 1985. ISBN 90-6144-155-2

(156) Amelsfort, A.M.J. van and Th. Scharten
A THEORETICAL STUDY OF THE ELECTROMAGNETIC FIELD IN A LIMB, EXCITED BY ARTIFICIAL SOURCES.
EUT Report 86-E-156. 1986. ISBN 90-6144-156-0

(157) Lodder, A. and M.T. van Stiphout, J.T.J. van Eijndhoven
ESCHER: Eindhoven SCHematic EditoR reference manual.
EUT Report 86-E-157. 1986. ISBN 90-6144-157-9

(158) Arnbak, J.C.
DEVELOPMENT OF TRANSMISSION FACILITIES FOR ELECTRONIC MEDIA IN THE NETHERLANDS.
EUT Report 86-E-158. 1986. ISBN 90-6144-158-7