MASTER

Diagnosing yield loss using specially designed random logic

Mels, A.

Award date:
2000

TU/e technische universiteit eindhoven

Section of Information and Communication Systems (ICS/EB)
Faculty of Electrical Engineering
ICS/EB 746

Master's Thesis

Diagnosing Yield Loss using Specially Designed Random Logic

Arjan Mels

Coaches:	ir. K. Veelenturf (Philips Semiconductors B.V.)
Supervisor:	prof.ir. M.T.M. Segers
Date:	November 2000

The Faculty of Electrical Engineering of the Eindhoven University of Technology does not accept any responsibility regarding the contents of Master's Theses

TU/e technische universiteit eindhoven
faculteit elektrotechniek

PHILIPS
Semiconductors – Nijmegen - MOS4YOU

Diagnosing Yield Loss using Specially Designed Random Logic

Arjan Mels

Supervisor: prof. ir. M.T.M. Segers *(Eindhoven University of Technology)*
Coach: ir. K. Veelenturf *(Philips Semiconductors B.V.)*

February - November 2000

Contents

1 Introduction

1.1 Context

At MOS4YOU (a Philips' Semiconductors IC Foundry in Nijmegen) a specially designed chip is used for speeding up the introduction of new IC production processes and for monitoring running processes. This dedicated chip is used for two reasons:

- Often no commercial products are available in the early stages of the process introduction.
- A dedicated chip can be designed to suit the specific needs of process introduction and monitoring, like size requirements and analysability.

The Veqtor (vehicle for process qualification and monitoring), as the chip is called, has to be representative for normal products to cover all types of problems that might arise during production. It therefore contains all types of commonly used modules: RAM's, ROM's, analogue circuitry and random logic (also called standard cell logic).

The chip also has to be easily analysable in order to find the cause of problems as quickly as possible. To be able to do this it has to be possible to zoom in on separate modules and even further into these modules. Therefore, the modules are separately accessible from the outside.

For diagnosing the cause of problems in the production process, the use of RAM and ROM is well established: their regular structure and the possibility to access every memory cell separately makes them relatively easy to analyse. However, it is also important to be able to analyse random logic, as this exhibits problems that do not exist for RAM and ROM.

Random logic as applied in normal applications is difficult to analyse because it is hard to pinpoint specific failing cells. This is due to the fact, that normal random logic has a logic depth (the number of logic cells between two externally accessible points) that is quite large. With a given set of test patterns, it is only possible to determine a list of cells that are the most likely cause of a failure.

To overcome the low analysability of normal random logic, a special type of random logic module (further called RDMC) was added to the Veqtor chip. This type of module contains random logic with a logic depth of one. As a result, all in- and outputs of the logic cells are externally accessible, which makes it possible to directly attribute an electrical failure to a specific cell or its interconnect.

Although the design of these modules and a basic test program for them were finished at the beginning of this project, the diagnosis methodology necessary to make good use of them did not exist.

1.2 Project Description

The development of such a methodology was the main subject of the graduation project described in this report. However, the project as it was defined consisted of two closely related sub-projects. The first sub-project consisted of devising a diagnosis methodology for the RDMC modules. The second sub-project consisted of proposing changes to the test methods or the design that improve the usefulness of the RDMC modules.

The diagnosis methodology would need to be able to relate the test results to process problems. The problems should be identified as precise as possible. It was suggested that it might be possible to determine the process layer (or the layout structure) that was causing problems by analysing statistical data. Also it was considered interesting to determine the location of a failure as precise as possible.

The deliverables for the first sub-project not only included the devising of the diagnosis methodology, but also the development of the supporting (software) tools.

In the second stage of the project, improvements to the test methods and / or the design of the module had to be developed. It was known that the current module design and test methods are not sensitive to certain categories of defects (resistive shorts and opens etc.), while these types of defects are expected to become increasingly important. Also some possibilities for enhancements might present themselves during the work on the first sub-project.

The deliverable for the second sub-project was a proposal for a new test approach or for an enhanced design.

MOS4YOU was in the first phases of introducing a new, 0.18μm process when work on this project started. The first Veqtor chips with the RDMC modules were just starting to come out of the line. Therefore the results of the first sub-project could immediately be tested and used. If it was found that additional or different test methods were needed, these could also be used immediately. Changes to the design would of course have to wait for a redesign of the entire Veqtor. This would most likely occur for the next process after the 0.18μm process.

1.3 Structure of this Report

Before going into the actual work on the project, the next two sections will provide necessary background information on the Veqtor chip, the RDMC modules and on the testing of random logic.

It was considered necessary to make some changes to the tests used before the work on the diagnosis methodology could start. This is described in Section 4. Also described in this section is the way in which detailed results about these tests can be logged.

Section 5 describes the diagnosis methodology that was devised together with the considerations leading to it. Also included in this section are some examples of the application of the methodology and a short description of the tools that were implemented to support the methodology.

Changes to the test methods and to the design are considered in Section 6 and Section 7 respectively. The chapter on test methods contains sub-sections on additional static patterns, current-based testing and timing-based testing. The chapter on the design considers a number of changes that would make the RDMC modules more suitable for their purpose.

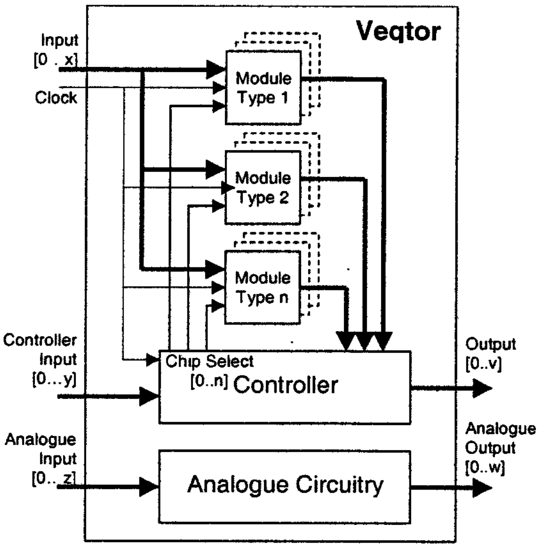
To finish, Section 8 provides a summary of the conclusions of the separate sections and the overall conclusions and recommendations.

2 The Veqtor & Random Logic

Before a detailed description of the RDMC module can be given in the next section, it is necessary to briefly look at the overall structure of the Veqtor chip. Details about the latest generation of Veqtor chips, which is used with the 0.18μm process, can be found in [Lit. 1]. This section also provides a basic explanation about random logic and how it can be tested and analysed.

2.1 Concept

The design of the Veqtor follows a modular approach (see the simplified structure in Figure 1). Consequently, different Veqtors can have different amounts and different types of modules. At the top level a controller switches the in- and outputs between multiple modules. In this way, the number of modules can vary while the number of in- and outputs can stay the same. The analogue circuitry is handled separately and it does not share its in- and outputs with the rest of the chip.



• Figure 1 Overview of the Veqtor

The contents of a module can be freely chosen, as long as the in- and outputs of a module adhere to the specification [Lit. 1]. Examples of module types that are used, are RAM, ROM and random logic. Multiple instances of each module are sometimes placed to increase the area (in order to increase the sensitivity for defects).

2.2 Versions

Currently, there are four generations of Veqtor chips. Each generation is targeted at a specific process. Within each generation, multiple versions of a Veqtor can exist. The different versions target different process options (for example with different amounts of metal layers) or contain different types and amounts of modules (because some were not available when the initial version was developed for example).

Because it is the first (and for now only) generation that contains the RDMC modules, the Veqtor4 is the only generation of interest for this report. The Veqtor4 is targeted at Philips' CMOS18 process (0.18μm and a maximum of 6 metal layers) and is used both for the start-up of this process and for the ramping of the yield.

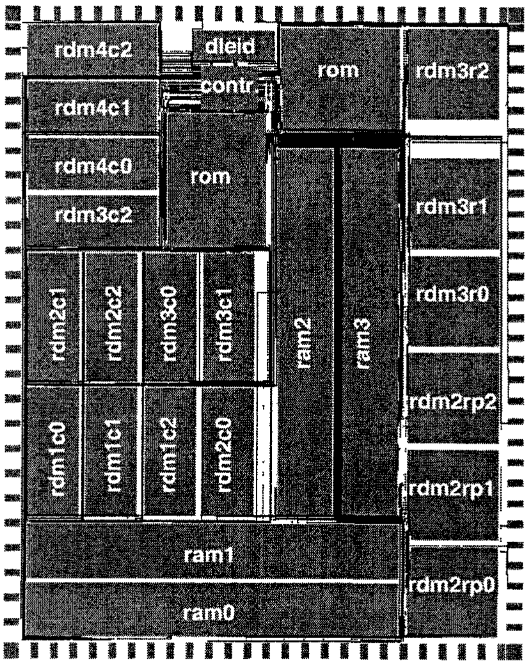
Although, the Veqtor4 is the first generation to contain the RDMC modules, random logic was already present on the earlier generations in the form of the so-called RDMR modules. These modules are derived from a 16-bit Digital Signal Processor. All RAM's and ROM's were stripped from the original design and all in- and outputs were made accessible via a scan-chain. The module is representative for a normal product, but difficult to analyse. Because it is so very representative, it is still maintained on the Veqtor4. More information about this module can be found in [Lit. 2].

All results used in this report are from three versions of the chip: the Veqtor4 Pinot, the Veqtor4 Flash and the Veqtor4 Steen. The Pinot and Flash differ only a little: the Flash is used for a process option with only five metal layers, so all modules with more than five layers are replaced with other variants. The Steen is a larger version, with more instances of each module type and in addition four RAM modules (these were not yet included on the Pinot and the Flash).

To get a feeling for the complexity of the chip, the layout of the Veqtor4 Steen is shown in Figure 2. The Veqtor4 Steen occupies an area of 31.5 mm^2 and contains about 12 million transistors.

2.3 Random Logic

Before starting to discuss .the random logic contained on the Veqtor chip, it is important to define what is understood by random logic. In addition, it is important to understand the basics of the testing of random logic because this clearly shows the problems in analysing it.



• Figure 2 The layout of the Veqtor4 Steen

2.3.1 Definition of Random Logic

The random in the name random logic, comes from the fact that it does not have a structure as regular as a RAM or a ROM. Random logic is —as already mentioned— also called standard cell logic. From this name, it can be implied that it is logic, built from standard cells, like AND-gates, OR-gates and flip-flops.

These standard cells are organised in what is called a standard cell library. (Therefore, the cells are sometimes also called library cells.) All cells in a library are designed in such a way that all their heights are equal and therefore they are easy to place in long rows. In addition, they are designed in such a way, that they only use the bottom one or two metal layers.

To define the logic functionality of a circuit, the standard cells are connected in the remaining metal layers according to the design. The way in which cells are connected, using the different layers of metal, is called the routing (also the name of the process of determining how the connections can be made). The more layers of metals available the more freedom available for the routing and the "easier" it is.

2.3.2 Testing Random Logic

In the introduction, it has already been postulated that it is difficult to isolate failures in normal random logic. This has to do with the way such a circuit can be tested, which will be briefly explained in this sub-section.

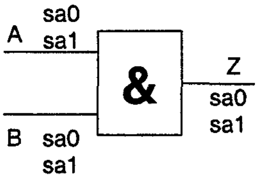
To determine if a chip functions according to the design, it has to be tested. A basic test strategy would be to insert the circuit into its final application and see if it functions. This of course is the ultimate test. Yet, this kind of test has a number of major drawbacks, not the least that it is very time-consuming and therefore costly.

The test method that is used most often in practise only tests the logic functionality of a chip; all other aspects (for example speed) are left out of consideration and therefore are not covered. For older processes, this proved to be no problem as short circuits are the predominant failure mechanism in this case. However, other types of defects (for example resistive shorts and opens) become increasingly important as feature sizes keep decreasing. These types of defects are not covered by the tests for logic functionality. Therefore additional test methods have been developed. Some of these are considered in Section 6.

Even when only the logic functionality of a chip is tested, it can become very time-consuming. In the case of a combinatorial circuit with N inputs, it would require 2^N test patterns to exhaustively test it. The test time can be strongly reduced by using the so-called stuck-at fault model. A stuck-at fault means that a certain net (connection between cells) is fixed to a certain logical value (either zero or one). On silicon, this would for example mean a short circuit between a net and ground. Two assumptions are introduced, which make the test time reduction possible:

- The only faults that can occur are stuck-at faults.
- Only one fault can occur in the circuit.

For example: for a 2-input AND-gate only three patterns are necessary (instead of four) to detect all possible single stuck-at faults (see Figure 3).

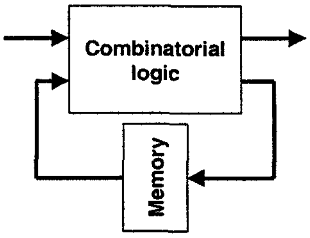


Test Pattern			Stuck-At Faults		
A	B	Z	A	B	Z
0	0	0			sa1
0	1	0	sa1		sa1
1	0	0		sa1	sa1
1	1	1	sa0	sa0	sa0

(The grey rows indicate the patterns needed to cover all stuck-at faults)

- Figure 3 Stuck-at faults for a AND-gate

The additional assumptions make that shorts between two nets (also called bridges) are not necessarily detected. This is caused by the fact that adjacent nets could have the same value during the entire test. However, it turns out that in practise the majority of the possible bridges is covered.

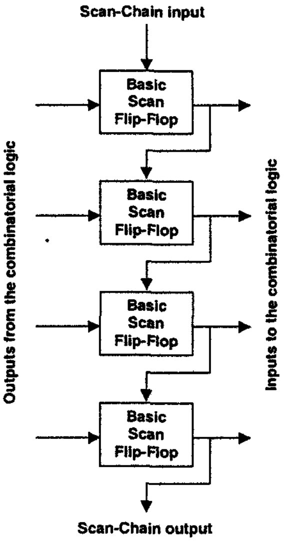


- Figure 4 Sequential logic

For sequential logic an additional problem occurs. Since the outputs of the memory (consisting of flip-flops) can be considered as additional inputs to the combinatorial logic (see Figure 4), the content of the memory has to be set to a specific state to test the logic. When the inputs of the memory are not directly connected to external pins, it might be impossible (or at least very difficult) to set it up in the correct way. In addition, it is also impossible to check all outputs of the combinatorial logic, when the memory cannot be read from the outside.

To overcome this problem, so-called scanable flip-flops (or just scan flip-flops) can be used. These flip-flops can have their value set via a dedicated input and they can be connected together into long chains (called scan-chains). These scan-chains actually form long shift registers, which makes it possible to set and read their content in a serial fashion (see Figure 5).

The use of scanable flip-flops reduces the testing to a combinatorial problem again. The price is a small increase in area because of the more complex nature of these flip-flops. The extra area needed is typically in the order of a few percents.



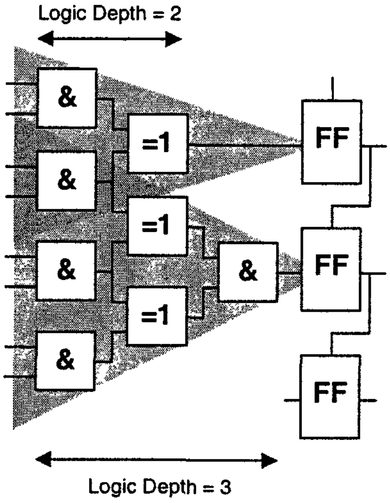
- Figure 5 Basic Scan-Chain

The use of scan flip-flops is a technique that is used industry wide and is an intrinsic part of the design flow used within Philips. Tools are available for the inclusion in the design of scan flip-flops and also for the generation of test patterns (Amsal). Amsal normally minimises the number of test patterns based on the stuck-at fault model.

2.3.3 Localising Defects

The testing of a circuit can tell if it is working properly. For real products, this is usually the only thing of interest. For diagnostical purposes, the cause of a failure has to be determined however. While testing a circuit, it is possible to pinpoint a specific scan flip-flop that contains a faulty value. However, the defect can be in all logic connected to the input of this flip-flop (see also Figure 6). Thus the fault is not localised precise enough yet.

It is possible to determine the location of the defect more precisely by combining data from multiple failing flip-flops and multiple test patterns, but usually this is still not enough to unambiguously pinpoint the location of a defect. See for example Figure 6. If the defect lies in the AND-gate that lies in the intersection of the two grey cones, then both flip-flops involved will indicate a failure (and therefore the failing gate can be pinpointed). However, in all other cases, only one flip-flop will fail and to establish the precise location of the defect, data from multiple patterns will have to be combined.



• Figure 6 Example of defect localisation

However, the normal procedure is to minimise the number of test patterns used, to keep the testing time low. It will thus not always be possible to pinpoint a single cell as the cause of a failure. Instead it might only be possible to give a list of potential defect locations. A tool (called faloc) is available within Philips to produce this list, but it does not provide very reliable results and work is in progress to improve the localisation capability.

3 Design of the RDMC Module

As already stated, it is very important to have random logic that is also analysable. It is simply a must-have to be able to solve process problems related to failures of the random logic. The RDMC module provides in this need. Although all details about the module can be found in [Lit. 1] and [Lit. 2], the most important points are summarised in this section.

3.1 Concept

The RDMC module was specifically designed to be analysable. The need for this high analysability became apparent from previous versions of the Veqtor, which only contained the RDMR modules. Although these RDMR blocks had some features added to make them easier to diagnose, it proved (almost) impossible to trace back the cause of failures.

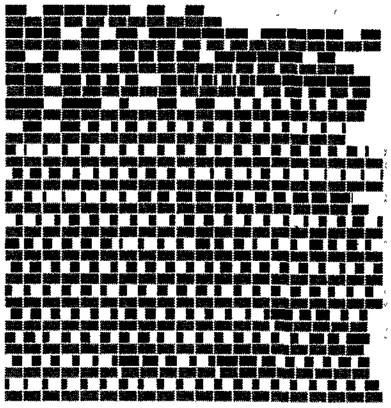
In the first place, the analysability of the RDMC modules is achieved by using logic with a depth of only one level. As could be seen in the previous section, this allows direct localisation of a defect given a failing scan flip-flop.

A second method, with which the analysability of the RDMC modules is increased, is using a couple of different variants. These variants have different amounts of routing in different metal layers. This should make it possible to determine if there are problems with a specific metal layer by analysing the differences in yield.

Although the RDMC module is not as representative for normal logic as the RDMR modules, it is still far more representative for normal logic than special test structures. Test structures are designed in such a way, that they are very sensitive to specific potential problems; they are of no use to detect unexpected problems.

3.2 Design

The logical function of the RDMC module could be freely chosen as long as the logic was one level deep. The obvious choice is to include one instance of each library cell. In this way, all possible configurations of the layers used in the cells are present. Therefore at least the front-end (in this report the front-end includes all layers unto and including metal1) is representative for normal random logic. Consequently, the module can also be used for the (partial) qualification of the standard cell library.



Dark grey cells are standard library cells
Medium grey cells are scan-chain flip-flops
Light grey cells are auxiliary cells

• Figure 7 Layout of the RDMC module

For each output of the standard library cells, a scan-chain flip-flop was added to the circuit; the inputs of the standard cells were connected to the closest scan flip-flop. The circuit was laid out with alternating rows of standard cells and scan-chain flip-flops (see Figure 7). Some buffers were added to provide sufficient drive-strength for the global nets.

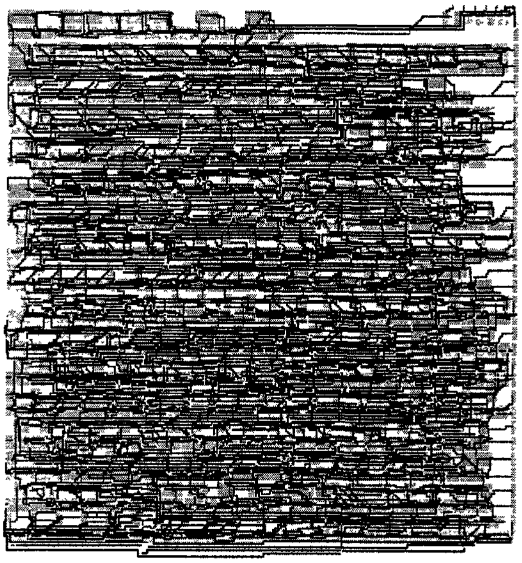
The inclusion of non-combinatorial cells, like flip-flops and bus-keepers, causes a problem however: they not only have to be tested for their combinatorial behaviour but also for their ability to retain data. If all inputs (including special ones like the clock input) of these cells would be connected to the outputs of the scan-chain flip-flops, it would be impossible to change these inputs independently. This would make it impossible to test all the functionality of these cells. Therefore, a couple of global nets are introduced (clock2, reset, preset and trig). The resulting circuit has 6 inputs (scan-in, clock, clock2, reset, preset, trig), and 1 output (scan-out).

3.3 Variants

Four different variants of the RDMC module were produced. The library cells are arranged the same, but the modules are routed differently.

The first variant uses the shortest-possible routing. This is the reference variant. Problems in the metal layers should only have a very small impact on this type. This makes it possible to determine if there are any front-end (including metal 1) problems.

The other variants have as much routing as possible in two of the metal layers (the layers differ for each of the variants). Thus problems with a specific layer have an impact that is as large as possible on the variant that uses this layer. The variants in layers 2 and 3 respectively in layers 4 and 5 minimise the number of vias in all layers except between 2 and 3 respectively 4 and 5. The variant with routing in layers 2 and 6 was intended to create a maximum amount of vias in all layers.



- Figure 8 A routed RDMC module (RDM4C)

The characteristics of the different modules can be seen in Table 1. The longer routing for variants 2, 3 and 4 was obtained by shifting all scan-chain flip-flops by two places in their respective rows (the last two ones were placed in front). This strategy gives routing that is about twice as long, but with a specific pattern of a couple of cells with very long wires and the rest with shorter wires (but still longer than in the original).

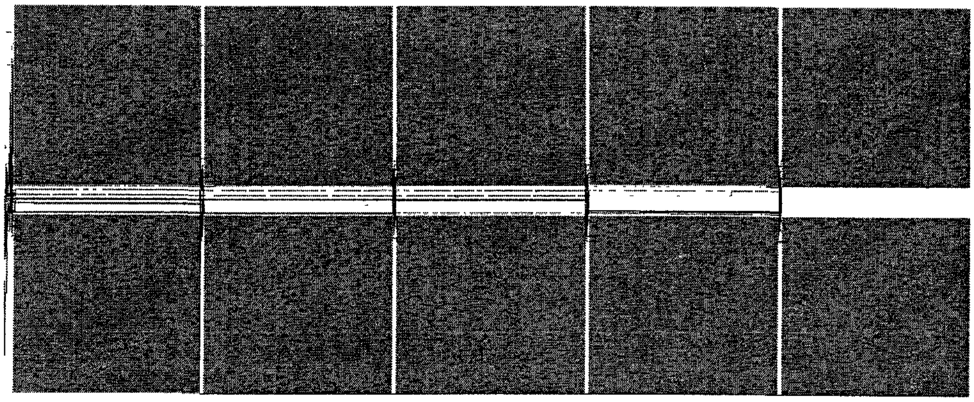
- Table 1 Routing Characteristics of the different RDMC's

	rdm1c	rdm2c	rdm3c	rdm4c
metal2*	16.6	20.5	2.2	2.6
metal3*	8.5	26.2	2.6	23.9
metal4*	0.4	0.4	23.9	1.6
metal5*	-	-	20.1	1.3
metal6*	-	-	-	17.8
# of via1	3038	3059	3030	3075
# of via2	1424	2212	3013	3141
# of via3	69	74	3792	1920
# of via4	-	-	2728	822
# of via5	-	-	-	951

(* metal lengths in mm)

3.4 Area Considerations

The area of all these variants is the same and about 0.05mm^2 (220μm^2 x 220μm^2). Because this area is too small to be effective, ten modules (of the same type) are placed together, with their inputs connected in parallel (see Figure 9). This has the additional benefit that the 10 modules can be tested in parallel, while keeping the number of necessary pins low. The resulting block will be called an RDMC module in the rest of this report and the composing blocks sub-modules. The area of the entire RDMC block (including some overhead for power connections etc.) is about 0.6mm². To increase the area even further multiple instances of each type are placed on the *Veqtor*.



- Figure 9 An RDMC module containing 10 sub-modules

4 Testing the RDMC module

During the testing of the Veqtor chip, data about the RDMC modules is collected. The diagnosis methodology that was developed and which will be explained in the next section, relies on this data. Although a test was implemented for the modules, this had to be improved to reduce the testing time. This section explains how this new test was constructed and how logging of detailed information is implemented.

4.1 The Tester

Most Veqtor chips are not packaged because they are not commercial products. So testing of the Veqtor is done on the wafers. The testing is done on a Schlumberger SX100 tester. This tester has strong failure logging capabilities, which makes it suitable to produce memory bitmaps and to log failures in random logic (see also Section 0).

All logic on the Veqtor is first tested for scan continuity. This test proves that the scan-chains are working properly by shifting a certain pattern through the chains and checking if this pattern comes out at the other side. After the scan-continuity test, the RDMC modules are tested with stuck-at patterns. All these tests are done under four conditions: 1800mV and 100ns cycle time (nominal), 1800mV and 50ns, 1950mV and 100ns, and 1650mV and 100ns.

4.2 Test Pattern Considerations

The RDMC modules are tested with stuck-at test patterns that test each cell exhaustively to have a coverage that is as high as possible. The presence of the non-combinatorial cells is a complication for the test pattern generation because it is necessary to test these cells for their data retention capability. This can only be tested by using two test patterns. The first pattern loads the data into the cell and the second pattern checks if the data is retained. This is not supported by the standard test pattern generation tool (Amsal) and therefore other ways to generate the patterns had to be explored.

Before the start of this project, a dedicated tool was chosen as the solution [Lit. 3]. The tool was implemented and patterns were generated with this tool. The drawback of this tool is that it tests the library cell sequentially. Therefore, the number of test patterns needed was very large and consequently the test time was very long.

The tool reads (a special version of) the netlist and a file with the test patterns for each cell. It combines the information from these two files and produces a file with the test patterns. The tool also generates a file which contains information about which cell is tested in which cycle. The information from this file can be used to determine the failing cell once the tester indicates that a module fails in a certain cycle.

4.2.1 Compact Test Pattern Generation

Although the pattern file generated with the tool is usable, it was decided that the number of resulting test patterns was unacceptably large. The testing of the RDMC modules took half of the total testing time for the Veqtor for example. Three possibilities for the reduction of the amount of patterns were identified:

- Enhance the custom pattern generation program to combine patterns where possible:
  - + Exact control over generated patterns
  - +/- Reasonable implementation time (1 to 2 weeks)

+/- Reasonably small pattern set

- Use Amsal after remodelling the non-combinatorial cells:
  - + Potentially the largest reduction in the number of patterns
  - - Correct testing of non-combinatorial elements uncertain, but is might be possible using delay fault patterns
  - - Implementation time unknown
  - +/- Use of standard tools (like aplot/faloc). However, delay fault patterns are not yet supported by these tools
- Use a combination of patterns generated by Amsal (for the combinatorial cells) and existing version of the pattern generation tool (other cells):
  - + Small amount of time needed for the implementation (1 to 2 days)
  - - Only the patterns for the combinatorial cells are reduced (estimated reduction to about 1/2)
  - - Combination of tools might give problems for the fault localisation

The enhanced custom program was reckoned the best option. This program has to combine the test patterns for multiple cells. The complicating factor is that the patterns for the non-combinatorial elements have to stay in the same order. This decreases the possible combinations of patterns. An additional optimisation with a factor two can be easily reached because the original program does not shift the data out and in simultaneously, but first shifts the data out (while shifting in zeroes) and only then shifts the data in. The program was implemented in the expected time span. More information about the program can be found in [Lit. 4].

4.2.2 Test Pattern Verification

The new pattern set that was generated with the tool is about 60 times smaller than the original set. To verify this new pattern set, it was used in parallel with the original set. The yield (ratio of failing to passing modules) was expected to be about the same for the two sets as they have the same coverage (for stuck-at faults).

However, the yield for the new set was about 1% to 1.5% higher than for the old set. This means that the coverage for the new set is lower. This yield difference was judged too large to use the new set. Three effects that could possibly cause the yield difference were identified:

1) The large pattern set should be better at detecting bridges because the environment of the cell that is being tested always has the same state. A bridge (a short between two signal nets) can only be detected if the two involved nets have opposite values. If the environment is always the same, the nets have opposite values in at least one situation; this is not necessarily the case with the smaller pattern set.
2) The activity level in the circuit is higher with the small pattern set because more cells change states simultaneously. This results in higher currents, which might cause other defects to be detected.
3) The shifting out and shifting in of the data is separated in the large pattern set, while this is done simultaneously for the small set. Although it is not directly obvious that this has an impact on the coverage, it is a difference between the two sets.

To establish whether the first two effects combined or the third effect caused the difference, additional pattern sets were generated. To test for the first two effects a set in which adjacent cells are not tested simultaneously was generated. To test for the third effect, sets with a different interval between shift out and shift in were used.

The results from all these tests can be seen in Table 2. The set with 3441 patterns is the original set, the sets with 99 patterns are the small sets and the sets with 408 patterns are the sets in which adjacent cells are not tested simultaneously. The number of zeroes used between shift out and shift in is the interval. The patterns with 0 interval simultaneously shift the data out and new data in; the scan-chain is 319 flip-flops long and thus the set with 319 interval successively shifts the data out and in. The others use half respectively double the interval. Only an indication of the yield differences is shown because the absolute yield differs between different RDMC modules and measurements from two separate runs are used. The yield of the original set was chosen as reference and set to 0.0%.

- Table 2 Indicative relative yields of pattern sets

		Patterns		
		3441	408	99
Interval	0		+1.3%	+1.3%
	160			+0.6%
	319	0.0%	±0.0%	±0.0%
	638			-0.5%

It can be seen from Table 2 that the number of patterns has a relatively small influence. This seems to indicate that the activity level (effect 2) and the bridging defects (effect 1) have no impact at all or an equal (but opposite) impact. However, the simultaneous or successive shift out and shift in has a large impact and concurs with the original 1% to 1.5% difference. To study this effect in more detail additional pattern sets were generated with variations for overlap of shift out and shift in (which use the smallest pattern set). The results from these tests can also be seen in Table 2.

To check, whether the content of the padding between shift out and shift in has any influence on the coverage, additional pattern sets with random instead of zero padding were generated. Two levels of random padding were tried. The first level only replaces the zeroes between shift out and shift in with random data. The second level also replaces the undefined values in the patterns themselves with random instead of zero data. Only the small sets with 99 patterns were modified.

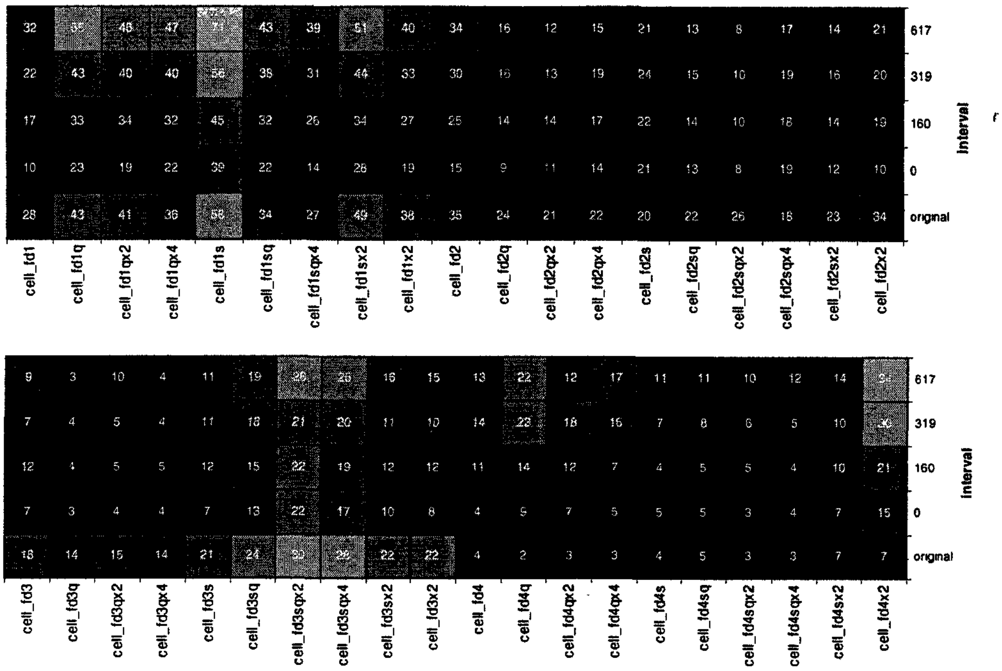
The coverage never is higher than with zeroes as padding, as can be seen in Table 3. For the simultaneous shift out and shift in no effect of the random padding can be detected. For the successive variant only a small difference is found when only the shifting is stuffed with random data, but a significantly larger effect is found when the patterns themselves are also stuffed. No explanation for this effect is obvious (as it does not occur with the simultaneous shift out and shift in).

- Table 3 Indicative relative yields of pattern sets

		Random padding		
		None	Shift	All
Inter- val	0	+1.2%		+1.2%
	319	0.0%	+0.2%	+1.0%

With the tools described elsewhere in this report, it was possible to establish which cells caused the yield differences. Figure 10 and Figure 11 (on the next page) show the number of failures per cell type for the different pattern sets. The number of failures is also indicated graphically by the intensity of the squares. (The cells that are not included in the figure have about equal failure counts for all pattern sets.)

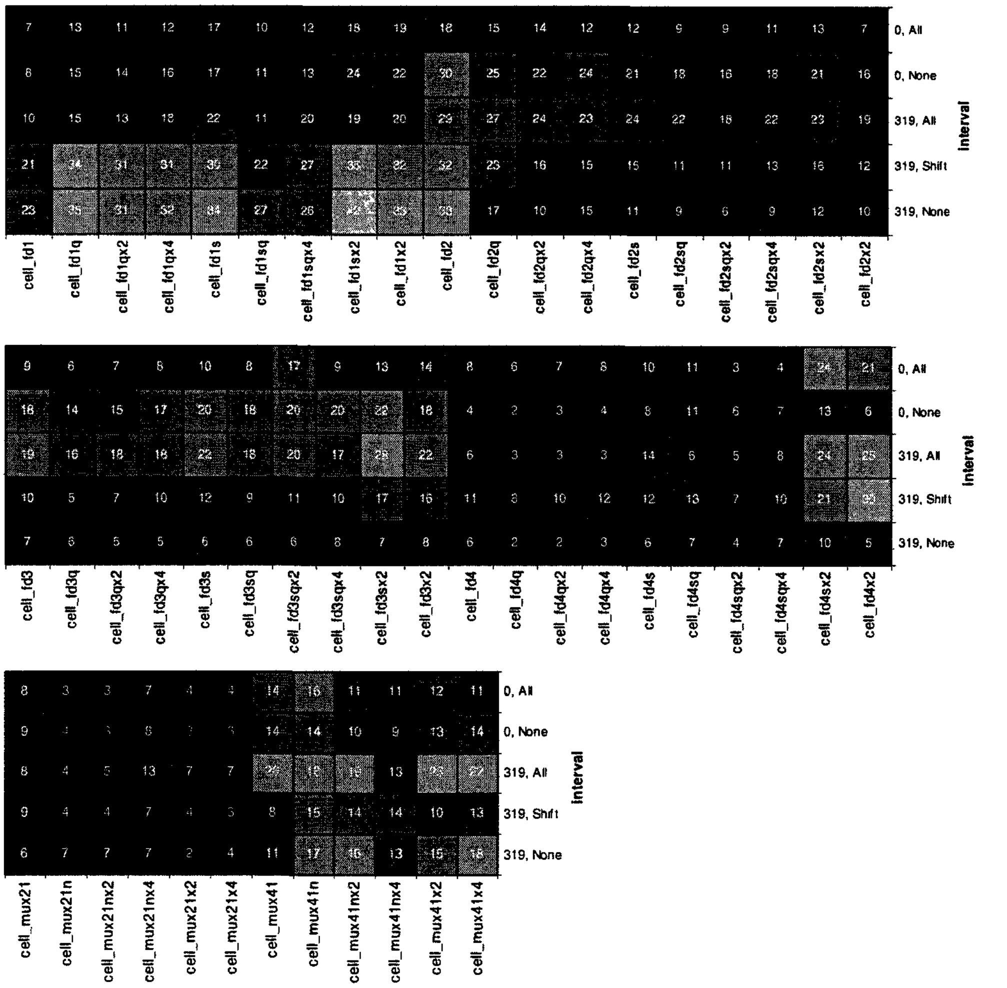
From Figure 10 it can be seen that certain types of flip-flops are responsible for the yield differences between the sets with different amounts of padding. For the insertion of random data, the results can be seen in Figure 11. The picture is not as clear as for the interval variation. Not only the flip-flops seem to play a roll in the yield differences, but also the multiplexers. Moreover, the patterns with successive shifting and no random padding and the patterns with padding in only the shifting, resemble each other. On the other hand, the one with successive shifting and complete random padding and the one with simultaneous shifting and no padding also resemble each other.



Interval	cell_fd1	cell_fd1q	cell_fd1qx2	cell_fd1qx4	cell_fd1s	cell_fd1sq	cell_fd1sqx4	cell_fd1sx2	cell_fd1x2	cell_fd2	cell_fd2q	cell_fd2qx2	cell_fd2qx4	cell_fd2s	cell_fd2sq	cell_fd2sqx2	cell_fd2sqx4	cell_fd2sx2	cell_fd2x2
617	32	55	48	47	71	43	39	51	40	34	16	12	15	21	13	8	17	14	21
319	22	43	40	40	56	38	31	44	33	30	16	13	19	24	15	10	19	16	20
160	17	33	34	32	45	32	26	34	27	25	14	14	17	22	14	10	18	14	19
0	10	23	19	22	39	22	14	28	19	15	9	11	14	21	13	8	19	12	10
original	28	43	41	36	58	34	27	49	38	35	24	21	22	20	22	26	18	23	34

Interval	cell_fd3	cell_fd3q	cell_fd3qx2	cell_fd3qx4	cell_fd3s	cell_fd3sq	cell_fd3sqx2	cell_fd3sqx4	cell_fd3sx2	cell_fd3x2	cell_fd4	cell_fd4q	cell_fd4qx2	cell_fd4qx4	cell_fd4s	cell_fd4sq	cell_fd4sqx2	cell_fd4sqx4	cell_fd4sx2	cell_fd4x2
617	9	3	10	4	11	19	26	26	16	15	13	22	12	17	11	11	10	12	14	34
319	7	4	5	4	11	18	21	20	11	10	14	23	18	15	7	8	6	5	10	30
160	12	4	5	5	12	15	22	19	12	12	11	14	12	7	4	5	5	4	10	21
0	7	3	4	4	7	13	22	17	10	8	4	9	7	5	5	5	3	4	7	15
original	18	14	15	14	21	24	30	28	22	22	4	2	3	3	4	5	3	3	7	7

- Figure 10 Influence of different intervals on the amount of failures of flip-flops



Interval	cell_fd1	cell_fd1q	cell_fd1qx2	cell_fd1qx4	cell_fd1s	cell_fd1sq	cell_fd1sqx4	cell_fd1sx2	cell_fd1x2	cell_fd2	cell_fd2q	cell_fd2qx2	cell_fd2qx4	cell_fd2s	cell_fd2sq	cell_fd2sqx2	cell_fd2sqx4	cell_fd2sx2	cell_fd2x2
0, All	7	13	11	12	17	10	12	18	19	18	15	14	12	12	9	9	11	13	7
0, None	8	15	14	16	17	11	13	24	22	30	25	22	24	21	18	16	18	21	16
319, All	10	15	13	18	22	11	20	19	20	29	27	24	23	24	22	18	22	23	19
319, Shift	21	34	31	31	30	22	27	35	32	32	23	16	15	15	11	11	13	16	12
319, None	23	35	31	32	34	27	26	[illegible]	35	33	17	10	15	11	9	6	9	12	10

Interval	cell_fd3	cell_fd3q	cell_fd3qx2	cell_fd3qx4	cell_fd3s	cell_fd3sq	cell_fd3sqx2	cell_fd3sqx4	cell_fd3sx2	cell_fd3x2	cell_fd4	cell_fd4q	cell_fd4qx2	cell_fd4qx4	cell_fd4s	cell_fd4sq	cell_fd4sqx2	cell_fd4sqx4	cell_fd4sx2	cell_fd4x2
0, All	9	6	7	8	10	8	17	9	13	14	8	6	7	8	10	11	3	4	24	21
0, None	18	14	15	17	20	18	20	20	22	18	4	2	3	4	8	11	6	7	13	6
319, All	19	16	18	18	22	18	20	17	28	22	6	3	3	3	14	6	5	8	24	25
319, Shift	10	5	7	10	12	9	11	10	17	16	11	8	10	12	12	13	7	10	21	33
319, None	7	8	5	5	6	6	6	8	7	8	6	2	2	3	6	7	4	7	10	5

Interval	cell_mux21	cell_mux21n	cell_mux21nx2	cell_mux21nx4	cell_mux21x2	cell_mux21x4	cell_mux41	cell_mux41n	cell_mux41nx2	cell_mux41nx4	cell_mux41x2	cell_mux41x4
0, All	8	3	3	7	4	4	14	16	11	11	12	11
0, None	9	4	5	8	7	5	14	14	10	9	13	14
319, All	8	4	5	13	7	7	26	18	19	13	23	27
319, Shift	9	4	4	7	4	5	8	15	14	14	10	13
319, None	6	7	7	7	2	4	11	17	16	13	15	18

- Figure 11 Influence of random padding on the amount of failures of flip-flops

4.3 Logging

The process of storing the results of the tests is called logging. For commercial products, it is only logged if a module passes or fails. To gather data that is more detailed, it should also be logged which test vectors failed. From this data, it can be determined which scan-chain flip-flops contained faulty results. For the RDMC modules, this also directly indicates which standard library cell is involved (see Section 5.3).

The Schlumberger SX100 tester stores the cycle number of the vectors that fail into a special memory during testing. This memory also contains information about the pins that failed, which is used to determine which sub-module failed. The standard software of the tester offers an easy way to log which vectors fail via its data logging options. This option could not be used because it is neither stable nor fast enough to handle the large amounts of data involved.

To circumvent these problems custom logging routines were written; details about these routines can be found in [Lit. 4]. The logging routines are written in such a way that they can also be used for other types of random logic modules (like the RDMR). The custom routines are not only more stable and faster, but they also reduce the amount of data that needs to be written in a number of ways:

- The log is compressed as it is written to disk.
- Only the results from the modules, which pass the scan-continuity testing are logged.
- The results of a module are not logged if there are more than 1853 failing vectors. (There are 1854 failing vectors if the module has floating outputs.)

The stated measures not only reduce the necessary disk space, but also reduce the testing time. (The disk accesses to write the logs cost more time than the actual testing.) The total test time of the Veqtor chip is only increased by approximately 15%.

Over the last couple of months, about 30 lots were tested and the detailed failure data logged. However, because of some test problems (that are unrelated to the logging), not all these results are usable.

4.4 Conclusions

Because of the excessive test time required by the original test pattern set, a new set was produced. The results of the set with 99 patterns and successive shift out and shift in (interval of 319) without random padding are close enough to the original set. Therefore, this set replaces the original one for all testing of the RDMC modules.

The RDMC modules are quite sensitive to slight pattern variations. This is mainly because of the presence of flip-flops. It seems to be a problem with the retention of the data because longer intervals cause more failures. The values that are used for padding also seem to play a roll, but this interaction is not clear from the data presented here.

The pattern dependencies probably do not occur in the same way for normal random logic because no flip-flops are present outside the scan-chain. The problem with the data retention probably does occur however (see also [Lit. 5]) and therefore it is worthwhile investigating it on normal random logic.

The logging could be implemented in such a form that the data from all the devices that are tested could be logged. This was actually done for almost all lots from the time that the logging was implemented.

5 Diagnosis Methodology

After the preliminaries of the previous couple of sections, it is now time to really look at the problem at hand. For the first part of the project, a diagnosis methodology for the RDMC modules had to be devised. This section will describe the methodology that is developed and also —though briefly— the tools that are necessary to support it.

5.1 Outline

For the start-up of IC production processes and improving the quality of the processes, it is necessary to be able to find the causes of problems in random logic. Problems in the production process out themselves as devices that fail some tests. The fraction of devices that pass all tests is called the yield. Due to the complexity of the production process and the impossibility of maintaining an absolutely particle-free environment, the yield will never be 100%, although the goal is to come as close as possible of course.

The purpose of the diagnosis methodology that is described in this section is to identify process problems as precise as possible with the help of the data that is gathered from the RDMC modules. The diagnosis methodology will of course rely heavily upon the special characteristics of the RDMC modules: first, there are the four different routing variants and, second, there is the capability to easily establish which library cell is failing.

During the design of the RDMC modules, two ways of diagnosing problems were identified [Lit. 2]:

- Analysis of the yield differences between different variants.
- Failure analysis (which is the process of finding the cause of a specific defect by physical inspection).

Failure analysis is a very time-consuming procedure. Therefore, it should only be used as a last resort, when all statistical analysis methods have been tried. However, when failure analysis has become inevitable, it has to be facilitated as well as possible. This can be done by determining the most likely location of the defects as precise as possible based on the test data.

A method of analysis that was not considered during the design was using the information about failing cells in a statistical way. It requires the availability of a large amount of data, but it is possible to obtain this (as was seen in Section 0). The analysis of the cell level data can give very detailed and useful information and the methodology described further on in this section relies heavily on this data.

The next sub-section handles about the modelling of the relationship between process problems and the yield. The sections after this go into the diagnosis methodology: first about using yield differences between the different variants, next about relating the cell level data to process problems and finally about some additional analysis instruments. Section 5.6 briefly describes the tools that make the usage of the methodology possible.

5.2 Yield Modelling

The yield model that is described in this sub-section has a number of limitations. An important assumption for the model is that defects are randomly and uniformly distributed. This assumption does not necessarily hold for IC production process, especially not for new processes. While the largest problems with mature process are particle-related defects, new process also suffer from systematic and parametric problems. It is however, a model that is simple enough to be usable and it has proven to be sufficiently precise for most purposes.

The yield of a device can be estimated by multiplying the yields of its constituent blocks. This is true because the yield can be interpreted as the probability that a randomly selected block is functioning properly and for a device to work all the blocks have to work. The estimation can be expressed as:

$$Y_{\text{total}} = Y_{\text{block 1}} \cdot Y_{\text{block 2}} \cdot \ldots \cdot Y_{\text{block n}} = \prod_{\text{all blocks}} Y$$

The same sort of relationship is true for all the separate processing steps. For a device to work properly all processing steps have to be flawless. Therefore, the total yield can again be expressed as the product of the constituent yields:

$$Y_{\text{total}} = Y_{\text{step 1}} \cdot Y_{\text{step 2}} \cdot \ldots \cdot Y_{\text{step n}} = \prod_{\text{all steps}} Y$$

If it is assumed that yield loss is caused by defects, it is possible to model the yield as the probability that a certain device contains no defects. If the defects are distributed randomly and uniformly, the probability that a chip contains a certain number of defects can be modelled by the binomial distribution [Lit. 6]. If there are n defects distributed amongst N devices, the probability P_k that a certain device contains k defects is:

$$P_k = \frac{n!}{k!(n-k)!} \cdot \left(\frac{1}{N}\right)^k \left(1 - \frac{1}{N}\right)^{n-k}$$

If n and N are large, it is possible to approximate this equation by the Poisson distribution (for proof see for example [Lit. 8]):

$$P_k \cong e^{-n/N} \frac{(n/N)^k}{k!}$$

If A is the area of the device and D_0 the defect density, the number of defects can be calculated as:

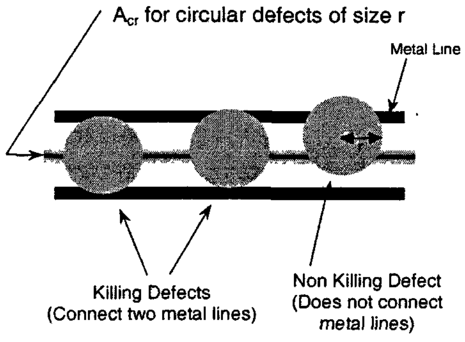
$$n = N \cdot A \cdot D_0$$

The probability that a device contains no defects, which is the yield, can be expressed as:

$$Y = P_0 = e^{-n/N} = e^{-AD_0}$$

This model is called the Poisson model and has the following limitations:

- Only random defects are accounted for: systematic and parametric yield losses are not modelled.
- A uniform distribution of the defects over the wafer is assumed, so clustering of defects is not taken into account. (For models that do take this into account, see [Lit. 6].)
- It is assumed that every defect actually causes a failure: this is not the case, but depends on both the size of the defect and the actual layout of the device (see Figure 12 for an example).

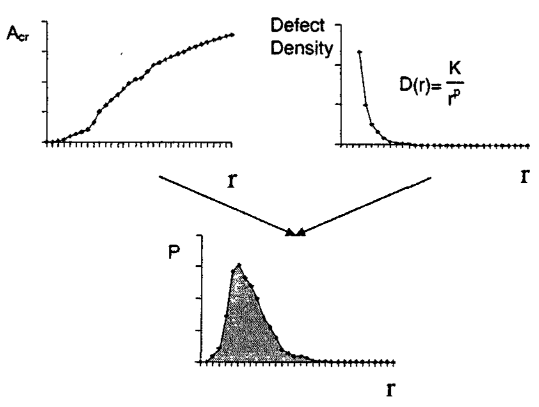


• Figure 12 Critical area and (non-)killing defects

An important extension to the Poisson model compensates for the fact that not all defects are killing. Extra material only causes a short if it connects two adjacent nets; missing material only causes a problem if the electrical connectivity is compromised. For current processes the two most important failure mechanisms (which therefore need to be modelled) are identified as:

- Shorts caused by extra material in layers with conducting lines;
- Opens caused by missing or bad vias and contacts.

In case of shorts, the Poisson model can be enhanced by using the so-called critical area model. The critical area of a chip is the area that is susceptible to a certain type of defect of a certain size (see again Figure 12). This means that the critical area is different for defects of different sizes. Therefore, the average number of defects per device (AD$_0$) is replaced by an integral over the defect size of the product of the critical area and the defect size distribution (see Figure 13 for a graphical example):



- Figure 13 Critical area and defect density distribution

$$Y = e^{-\int_0^\infty A_{Cr}(r)D(r)dr}$$

The defect density curve D(r) is never exactly known. Only the values at a limited number of sample points can be obtained. However, according to [Lit. 7] it is possible to model D(r) using:

$$D(r) = \frac{K}{r^p}$$

In this equation, K describes the defect density level and p is a measure for the form of the distribution of the defect sizes.

The critical area A$_{cr}$(r) is also only known at a couple of points. No simple model can be used in this case, so the curve is approximated with a piecewise-linear function. If $r_1 \ldots r_N$ are the radii at which A$_{cr}$(r) is sampled, the entire function can now be approximated as:

$$Y = e^{-\sum_{n=1}^{N-1} \frac{D(r_n)A_{cr}(r_n)+D(r_{n+1})A_{cr}(r_{n+1})}{2}\cdot(r_{n+1}-r_n)}$$

For evaluating the effect of open vias and contacts a simple equation can be used, in which N$_{via}$ is the number of vias in the device and P$_{fail}$ is the chance that a certain via is open:

$$Y = e^{-N_{via}P_{fail}}$$

With these equations the total yield can be modelled as:

$$Y_{\text{total}} = \prod_{\text{all blocks}} Y_{\text{block n}} = \prod_{\text{all blocks}} \prod_{\text{all steps}} Y_{\text{block n, step m}}$$

$$= \prod_{\text{all blocks}} \left(\prod_{\text{all shorts sensitive steps}} e^{-\int_0^\infty A_{Cr}(r)D(r)dr} \cdot \prod_{\text{all open sensitive steps}} e^{-N_{via}P_{fail}} \right)$$

The design attributes (critical area and number of vias) have to be obtained to be able to use the model. Special software is available to extract these attributes from the layout of a circuit. To use the model for yield prediction purposes, the defect density curves or probabilities of failure for all process steps have to be known. These could be obtained by specially designed test structures or by optical defectivity measurements for example.

However, when the purpose is to find out which process steps are problematic, the yield is known and the defect densities and failure probabilities are unknown. It is impossible to determine the defect densities and failure probabilities from just one yield equation of course. Therefore, it is necessary to obtain a set of equations.

5.3 Yield Differences between Modules

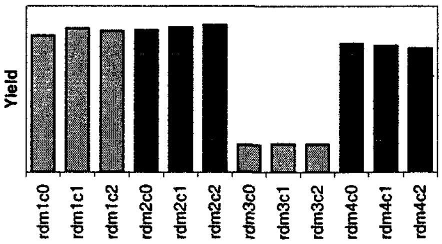
Using multiple modules with only small differences in the design is a method to obtain such a set of equations. This is the idea behind the different variants of the RDMC module. They were specifically designed to differ only in the metal and via layers (see Table 1 on page 12). The different types of modules normally would yield about the same. So if there are yield differences between the types, these can be attributed to a circumscribed set of layers.

There are a number factors that make interpreting the obtained yields difficult:

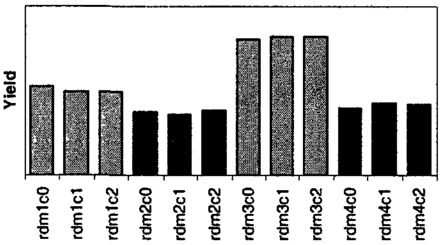
- There are only four different types of modules, but more metal and via layers. Therefore, the equations cannot be solved exactly and the finest distinction that can be made is that a certain set of layers is problematic.
- Because the modules are small and not very densely routed, the critical areas are quite low. The yield differences are therefore often only very small and disappear in the noise margins.
- There are a number of other factors that can also cause yield differences between different modules:
  - Modules are aligned differently (horizontal versus vertical).
  - Modules are connected to different power supplies.
  - Modules are placed in different locations causing their environment to differ.

Because of these reasons, it will not always be possible to precisely determine the cause of yield differences. However, if yield differences occur, they do give a clear signal that there are problems of some sort and the differences might give a starting-point for further investigations. The interpretation of the results should be done with great care and no (small) fixed set of rules can be given.

Some examples give a good idea of the possibilities of an analysis of the yield differences. On lot CZ6233, the low yield of the rdm3c modules indicates problems in either metal 4, metal 5 or via 4. This is supported by the evidence from special test structures, which indicate massive shorts in metal 4. In the case of lot CZ9893 however, the test structures indicate large problems with via 3 (and also some minor problems on metal 2). It can therefore be expected that again the rdm3c modules would show relatively low yields. However, rdm3c yields relatively high instead. This indicates that some other mechanism, which is not modelled, is important in this case. This cannot be seen from the measurements on the test structures and the RDMC modules prove their worth in this case.



• Figure 15 Indication of the yields of lot CZ6233



• Figure 14 Indication of the yields of lot CZ9893

5.4 Cell Level Differences

It is possible to go one step further than the block level differences. For this, it is necessary to determine the yield (which boils down to the number of failures) of each type of cell used in a module. The first step is of course to determine which cells cause the failures, the second step is to calculate the total number of failures for each cell type.

5.4.1 Determining the Failing Cells

For normal random logic, it is an involved process to determine which cell(s) caused a failure (see also 2.3.3). However, due to the special nature of the RDMC modules, it is possible to find the failing cells easily.

It can be determined which test patterns failed from the failing vector log (see Section 0). The first step is to translate the failing vector into the scan flip-flop that contained the faulty result. This is a simple one-to-one mapping: the vector number determines the position in the scan-chain of the failing flip-flop. The next step is to determine which standard library cell is connected to the input of this flip-flop. This is again a one-to-one mapping and this information can be extracted from the netlist of the circuit.

In the process of determining the failing cell, it is assumed that the failure is not caused by the scan-chain flip-flop itself. This is a safe assumption because almost the entire functionality of these flip-flops is already tested with the scan-continuity test. Therefore, a failure has to be caused either by the standard library cell or by the nets connecting the cells together.

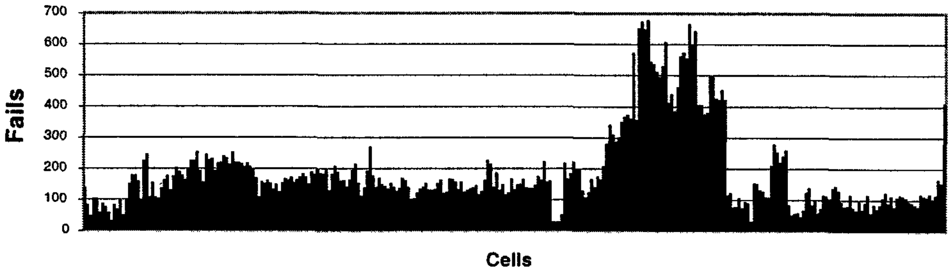
5.4.2 Counting the Failures

When it has been determined which cell(s) failed, the number of failing cells of each type per wafer and per lot can be calculated. The following points have to be taken into consideration:

- It is important to count every failing cell only once for each sub-module instead of counting each failing vector caused by this cell. (Otherwise, cells with many test patterns would be counted more often than cells with fewer patterns and cells with more outputs would also be counted multiple times.)
- Defects in the global nets (clock, preset, reset etc.) outside a sub-module should not cause the affected cells to be counted as defective. Such defects cause all cells that use this net, in all sub-modules connected to the net, to fail. This would cause a large failure count for these cells, but all due to a single defect. To exclude the affected cells from the counts, the modules for which all sub-modules have the same cells that fail should be excluded in their entirety. (Because this is difficult to establish in practise, the modules that have failure counts for all sub-modules that are within 10% of each other are excluded).

The failure counts are actually calculated separately for each sub-module and each operating condition. Also available are totals per module type per operating condition etc. etc. and totals for an entire lot. The data per wafer is not very useful due to the low counts. A short calculating exercise will clarify this. A wafer contains about 200 devices with 120 RDMC sub-modules each. This means that there are about 24000 instances of each library cell on a wafer. If the yield on the cell level were to be 99.9% (which gives a yield on an entire sub-module of about $0.999^{600} \approx 55\%$), this would mean a total failure count of on average 24 failures per cell type per wafer.

A graphic representation of some sample data can be seen in Figure 16. The number of failures (on the vertical axis) for each cell type (on the horizontal axis) is plotted. The cells are more or less grouped together according to function. The large peak contains all the flip-flops for example.



- Figure 16 Failures per cell of rdm4c on CZ9893

This kind of plot can already be useful by itself. Cells that fail conspicuously often can point to a design or process problem. The data can also be used to compare lots (or parts of lots) with different processing, to see if certain types of cells have different failure rates. If problems are seen, it is possible to zoom in further on these issues; more about this in Section 5.5.

5.4.3 Design Parameter Extraction

Apart from detecting obvious aberrations, it should be possible to make statements about which layers are problematic by comparing failure counts of different types of cells while taking the differences in their design into account. In order to do so, it is of course necessary to extract the relevant design attributes.

The attributes have to be extracted per cell. As already mentioned, it is not really determined that the cell itself is failing, but only that either the cell or one of the nets connecting to it is defective. It is therefore necessary to extract both the design attributes of the cell and of the connecting nets.

The extraction of the design attributes of the cell itself can be done by using the extraction tools, which are available, on each cell separately. (A small error in critical areas is made in this way because the critical area also is influenced by the neighbours of a cell, but this is deemed acceptable.) The tool extracts a number of different types of design attributes. This includes critical areas, contact (via) counts, but also transistor counts and edge lengths. Although some of these are not used in the current yield model, they could be included in a way very similar to the vias. For a complete list of all the relevant parameters that are extracted see Appendix A.

For metal layers 2 to 6 and the via layers the extraction tool is not (yet) usable because the tool does not have the capability to extract parameters per net. Therefore, a custom program was modified to extract these parameters. However, the implementation of a true critical area extraction capability is complicated and was outside the scope of the project. As best alternative, the length of the nets in each layer and the via counts of these nets are extracted. (See Appendix A for the parameter names.)

5.4.4 Yield Model for the Standard Library Cells

Now that the parameters that can be extracted are known, a model that is as complete as possible can be made. For cells, it is easier to speak about the number of failures than about the yield. It is possible to model the number of failures of a cell N_{fails} by using the failure rate R_{fails} like:

$$N_{\text{fails}} = N_{\text{cells}} \cdot R_{\text{fails}} = N_{\text{cells}} \cdot (1 - Y_{\text{cell}})$$

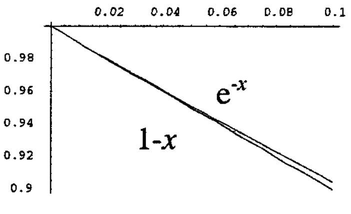
A number of different types of parameters are extracted: critical areas, coverage areas, counts and lengths. Except for the critical areas they can all be handled in the same way: the area, count or length times a probability of failure per unit gives the average number of failures. If the areas, counts or lengths are call $N_{\text{attribute}}$, the yield of a cell can be modelled like:

$$Y_{\text{cell}} = e^{-\left(\sum_{CA}\int_0^\infty A_{Cr}(r)D(r)dr + \sum_{\text{misc}} N_{\text{attribute}}P_{\text{fail}}\right)}$$

$$\cong e^{-\left(\sum_{CA}\sum_{n=1}^{R}\frac{D(r_n)A_{cr}(r_n)+D(r_{n+1})A_{cr}(r_{n+1})}{2}(r_{n+1}-r_n)+\sum_{\text{misc}} N_{\text{attribute}}P_{\text{fail}}\right)}$$

The yield of a single cell is very close to 100%, therefore it is possible to linearise this equation by using (see also Figure 17):

$e^{-x} \cong 1-x$ for small values of x



• Figure 17 Linearisation of e^{-x}

The model for the number of failures now becomes:

$$N_{\text{fails}} = N_{\text{cells}}\cdot(1-Y_{\text{cell}}) =$$

$$\cong N_{\text{cells}}\cdot\left\{\sum_{CA}\sum_{n=1}^{R}\left(\frac{D(r_n)A_{cr}(r_n)+D(r_{n+1})A_{cr}(r_{n+1})}{2}\cdot(r_{n+1}-r_n)\right)+\sum_{\text{misc.}} N_{\text{attribute}}P_{\text{fail}}\right\}$$

It is possible to rewrite this in the form:

$$N_{\text{fails}} = N_{\text{cells}}\cdot(1-Y_{\text{cell}}) =$$

$$\cong N_{\text{cells}}\cdot\left\{\sum_{CA}\left(\frac{(r_2-r_1)D(r_1)}{2}\cdot A(r_1)+\sum_{n=2}^{R-1}\frac{(r_{n+1}-r_{n-1})D(r_n)}{2}\cdot A(r_n)+\frac{(r_R-r_{R-1})D(r_R)}{2}\cdot A(r_R)\right)+\sum_{\text{misc.}} P_{\text{fail}}N_{\text{attribute}}\right\}$$

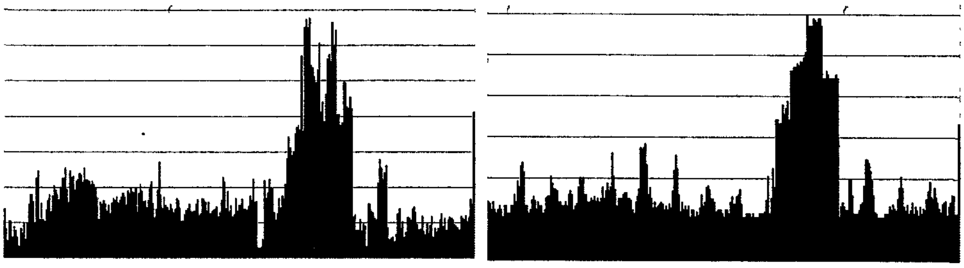
Note that this is an equation of the form:

$y = b + a_1x_1 + a_2x_2 + \ldots + a_nx_n$

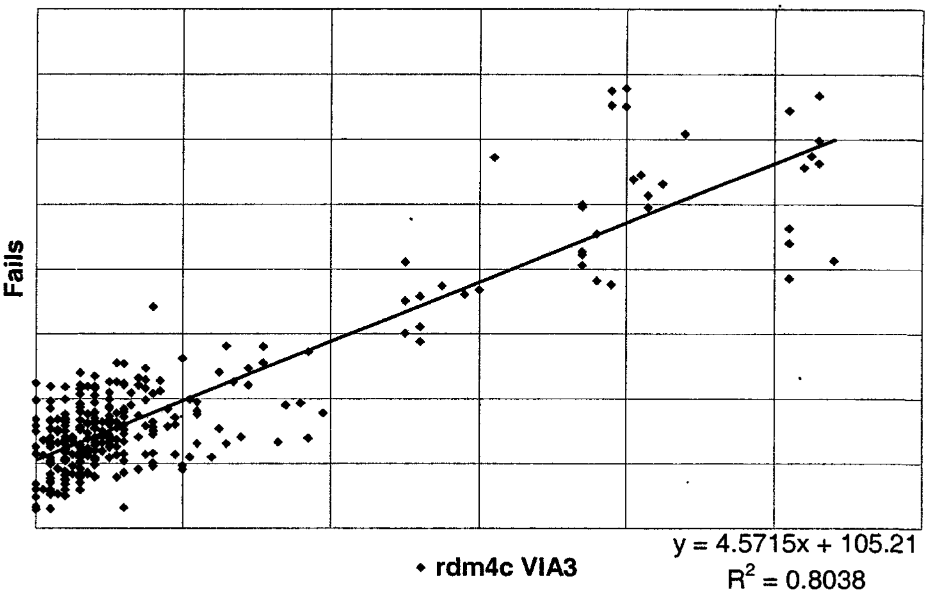
In this formula the x's are the design features $A_{cr}(r)$ and $N_{\text{attribute}}$ scaled by the number of tested cells N_{cells} and the a's are (a scaled version of) the defect densities D(r) and the probability of failures P_{fail}. The b would represent a basic level of yield loss. Of course, the values of all variables may only be positive to represent physically possible values.

5.4.5 Correlating Failures and Design Attributes

Because of the linear nature of the relationship, the design attributes can directly be correlated with the failure counts. The design attributes that correlate best have the largest impact on the failure counts and are therefore the ones that are most likely involved in excessive yield losses. Take for example the failure count on rdm4c on lot CZ9893 and the via 3 count for the rdm4c (see Figure 18). These values can be plotted against each other (Figure 19). It can be seen that the correlation is quite good.



- Figure 18 Measured failure counts (left) and extracted via 3 counts (right) for lot CZ9893



- Figure 19 Correlation between failures and via 3 on rdm4c of lot CZ9893

Instead of visually comparing the graphs for all design attributes, the correlation coefficients (R) can be calculated. The correlation coefficient squared (R^2) tells how much of the variation in one variable can be explained by the second. For the example, this means that about 80% of the variation in the failure count can be explained by via 3. (See for an extended example Table 4.)

- Table 4 Correlations (R^2) with metal for lot CZ9893

	Area	M1	M2	M3	M4	M5	M6	VIA1	VIA2	VIA3	VIA4	VIA5
Total rdm1c 1800mV 100ns	0.32	0.04	0.60	0.81	0.84			0.02	0.78	0.83		
Total rdm2c 1800mV 100ns	0.38	0.04	0.59	0.22	0.75			0.03	0.68	0.73		
Total rdm3c 1800mV 100ns	0.28	0.00	0.26	0.30	0.22	0.02		0.18	0.25	0.30	0.13	
Total rdm4c 1800mV 100ns	0 46	0.01	0.81	0.82	0.81	0 38	0 06	0.17	0.83	0.80	0.50	0.33
Average	0.36	0.02	0.57	0.54	0.65	0.20	0.06	0 10	0.64	0.67	0.32	0.33

By inspecting the correlation coefficients, a set of design attributes can be selected that each separately explain the failure counts to a certain degree. Since the design attributes are correlated themselves, it is impossible to determine how much of the failure counts can be explained by the combination of design attributes.

5.4.6 Selection and Linear Regression of the Design Parameters

Instead of looking at the separate design features, it is better to look at a combination of design parameters. It is not possible to solve the obtained yield equations exactly (because the problem is over-determined). However, a best fit for the constants can be obtained by using linear regression. The most commonly used criterion for this fit is that of the least squares. In this case, the purpose is to minimise the sum of squares of the difference between the actual values and the estimated values. (For the calculations involved in linear regression, see for example [Lit. 8].)

Although the technique could simply be applied to all the design variables, this will not yield very usable results. This is due to the correlations between the design attributes, which causes two problems:

- The constants that are calculated for a group of highly correlated variables have a very high degree of uncertainty and small changes in the data can cause large fluctuations in these constants.
- Some of the constants might become negative. (It is impossible for a design attribute to have a positive impact on the yield.)

Hence, it is necessary to find a subset of the design attributes that can explain the failure counts almost as well as the complete model can, but without using negative constants and without including highly correlated design parameters. This process of automatically selecting the variables that explain the actual failure counts best can be approached in a number of different ways:

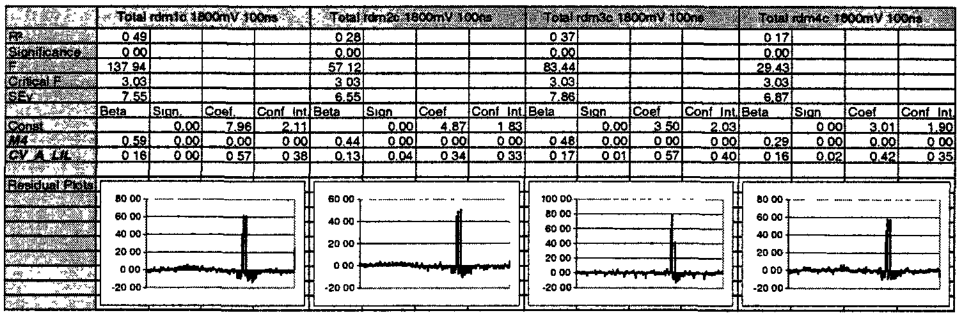
- Full subset search: Each possible subset of the design variables will be examined in turn. The smallest subset that has a correlation coefficient that is close enough to that of the full module will be chosen.
- Backward elimination of variables: Variables are removed from the full model until the correlation coefficient is not close enough to that of the full model any more. The removal can be based on several criteria: lowest explaining power, lowest statistical relevance etc.
- Forward inclusion of variables: Variables are included one by one (starting with the one that correlates best) until the correlation coefficient is sufficiently close to that of the full model.
- A combination of forward inclusion and backward elimination (also called stepwise selection): this is only useful if different criteria are used for inclusion and elimination.

All of the above mentioned methods have there merits and drawbacks. Full subset search is out of the question due to the large number of design attributes. There are just too many possible subsets and therefore the computation time would get to long. Although backward elimination is superior according to [Lit. 9], it poses a practical problem: Microsoft Excel, which is used to implement the analysis, only supports linear regression of up to 16 variables. This leaves forward inclusion and a combination of forward inclusion and backward elimination as options. The use of a combination did not seem to warrant the extra trouble. Therefore, the forward inclusion scheme was chosen.

For the forward inclusion scheme, it has to be decided which variable will be included next and when to stop including variables. In the implementation chosen, both are based on the correlation between the design attributes and the residual failure counts (the actual failure counts minus the estimation obtained with the variables already included). The variable with the highest positive correlation will be included. Since only variables with positive correlation coefficients are included, the constants will also be positive. No further variables will be included if the highest correlation coefficient is lower than a certain limit.

The scheme is complicated by the fact that there are 4 different variants of the RDMC module. These variants have different design attributes for the upper metal and via layers. Therefore, the correlations have to be calculated separately for each of the variants. For the selection and stop criteria, the average of the correlation coefficients is used.

An example of the output data can be seen in Figure 20. The most important information is of course the list of chosen design attributes. The correlation coefficient squared (R^2) indicates how much of the variation (and thus of the yield loss) is explained by the included design parameters. Per design attribute the calculated constant is included (Coef.), but also a normalised version of the constant (Beta). These normalised constants indicate the relative importance of the chosen attributes. In addition information is included on the statistical relevance of the total model and the separate design attributes (Significance, which should be lower than 0.05 for a 5% confidence interval). Finally plots of the residuals (the actual failure counts minus the estimated ones) are included.



	Total rdm1c 1800mV 100ns				Total rdm2c 1800mV 100ns				Total rdm3c 1800mV 100ns				Total rdm4c 1800mV 100ns			
R²	0 49				0 28				0 37				0 17			
Significance	0 00				0.00				0.00				0.00			
F	137 94				57 12				83.44				29.43			
Critical F	3.03				3 03				3.03				3.03			
SEy	7.55				6.55				7.86				6.87			
	Beta	Sign.	Coef.	Conf Int.	Beta	Sign	Coef	Conf Int.	Beta	Sign	Coef	Conf Int.	Beta	Sign	Coef	Conf Int.
Const		0.00	7.96	2.11		0.00	4.87	1 83		0.00	3 50	2.03		0 00	3.01	1.90
M4	0.59	0.00	0.00	0 00	0.44	0 00	0 00	0 00	0 48	0.00	0 00	0 00	0.29	0 00	0.00	0 00
CV A LIL	0 16	0 00	0 57	0 38	0.13	0.04	0 34	0 33	0 17	0 01	0 57	0 40	0 16	0.02	0.42	0 35
Residual Plots																

- Figure 20 Example of the output of the linear regression on lot CZ9701

5.4.7 Interpreting the output

The information obtained has to be carefully interpreted. Although the design attributes included model the failure counts best in a mathematical sense, this does not necessarily mean that they are actually causing excessive yield losses. Three things have to be kept in mind:

- Many of the design parameters are strongly correlated with each other:

  The design parameters that are included might only have fractionally higher correlations with the failure counts then some others. Still, when these parameters are included the strongly correlated ones will no longer be strongly correlated with the residues and none of them will be included. Therefore, it always has to be taken into account that either the indicated parameter or any of the parameters that are strongly correlated with it, might be the cause of the yield losses.

- Yield losses can be caused by design features that are not extracted:

  Therefore, the variables should explain the failure counts at least reasonably well (which can be determined by looking at the R^2). If this is not the case either no design parameters cause an excessive yield loss (the yield loss is spread amongst all design parameters) or the yield losses are caused by a feature that is not captured in a design parameter.

  The residual plots can be another indication that something that is not captured in one of the design attributes is causing yield losses. If the residual plots show a (strong) systematic pattern (as is the case in the example of Figure 20), this means that this pattern could not be explained by any of the extracted design attributes.

- If the stop limit is set too low, parameters are included that cause only normal yield loss, instead of excessive yield loss. It might also cause the inclusion of strongly correlated attributes, thus causing the statistical relevance of a number of parameters that were included earlier on to drop.

Even when all these considerations are taken into account, the method provides an indication of the problematic layout features and thus also of the problematic process steps.

5.4.8 Validation of the method

Although the method is theoretically sound, it should be verified in practise. To verify the method, the results are compared to the problematic features as found by special test structures (called YEM = Yield Evaluation Modules) for a couple of lots. The YEM's each target another layout feature; this includes shorts between conducting lines and opens due to vias. The parameters found with the analysis of the RDMC modules should correspond with the results of the YEM's.

The results of a comparison between the RDMC analysis and the YEM's can be seen in Table 5 (the detailed results for the RDMC analysis can be found in Appendix B). For lot CZ9700 there is a perfect correspondence. For lot CZ9701 the YEM's indicate that via 4 is problematic, but this could not be seen on the RDMC's. For lot CZ9893, there is a good correspondence as long as the correlations between the parameters are kept in mind: the critical areas for lil and poly are strongly correlated and also the via 3 count and metal 2 lengths are strongly correlated. Lot CZ8422 has numerous problems and the yield is low. Nevertheless, there is a correspondence between via 3 and a complex of via 2, via 3, metal 2 and metal 3 (which are all correlated). The RDMC module analysis shows that via 3 (or any of the strongly correlated parameters) can already explain about 50-80% (depending on the variant) of the yield loss. It looks like the other problems that can be seen on the YEM's are hidden by this overbearing problem.

- Table 5 RDMC analysis versus YEM's

	RDMC	YEM
CZ9700	CA_ M1_0015	M1
	CA_ LIL_0045	Lil-Lil
CZ9701	M4	M4 Opens
	CV_ LIL	Lil-Lil
		Via4
CZ9893	VIA3	M2 Via3
	CA_LIL_0085	Poly-Poly
CZ8422	VIA3	Via 2 & 3 M2 & 3
	NR_ACT_CNT	
		M6
		Lil-Act
		Poly-Poly
		Via1
		M1

The results for these four lots indicate that the correspondence between the RDMC analysis and the YEM's is quite good. This indicates that the method gives sensible results.

The residual plots of all these lots show (see Appendix B) to a certain extend the same pattern. Flip-flops of a certain type (fd2, fd2q, etc.) have a (much) larger residual than the other cells. This could be an indication of a problem that is not captured in a design attribute that is currently extracted. It is suspected that a certain combination of corners in the active layer is the cause. Investigations were ongoing at the time of the writing of this report however.

5.5 Additional Analysis Instruments

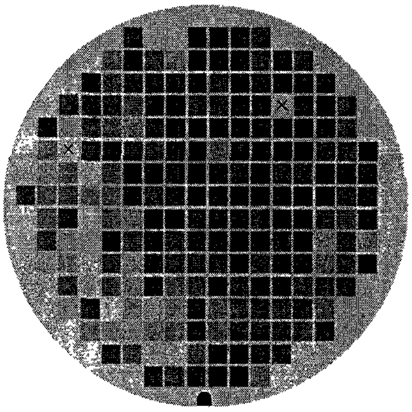
The correlation method described in the previous sub-section provides a good indication of the problematic design features. In many cases, this will already be enough. However, to determine exactly what is wrong with the indicated process steps further analysis is required.

Ultimately failure analysis might be necessary, but before this is attempted, there are still some more analysis instruments that might be helpful. Three such instruments will be described in this sub-section. No structured way has been devised to make use of them, but depending on the situation they can be used in various ways. In this section, the instruments as such are described and not (at least not exhaustively) the methods in which they could be used.

5.5.1 Wafermaps

Wafermaps are used to determine the spread over the wafer of a problem. This can give an indication if a problem is due to a waferstepper for example.

For the RDMC modules, separate wafermaps can be generated for each standard library cell (see for an example Figure 21). Because of the low failure counts, it is only useful to generate wafermaps for an entire lot and with all module types taken together.



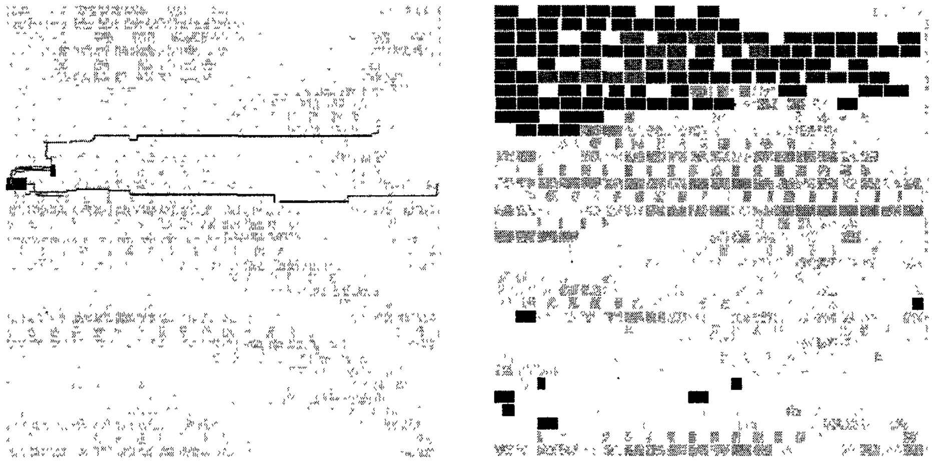
- Figure 21 Wafermap of a specific cell

5.5.2 Bitmaps

As was seen in Section 0 it is determined per sub-module which cell(s) fail. For failure analyses the co-ordinates of these cells within the die are needed. To obtain these co-ordinates it is necessary to translate the co-ordinates within the RDMC sub-modules to co-ordinates at the die level.

It also has to be taken into account that the defect does not necessarily lie within the cell, but could also lie in the connected routing. Although it is possible to calculate the co-ordinates of all the stretches of routing this is not practical. Therefore the software can generate bitmaps (like that in Figure 22) of the failing submodules. This together with the co-ordinates of the failing cell provides enough information to steer the physical localisation of the affected cells/nets for failure analysis.

Not only bitmaps per sub-module can be generated, but also so-called stacked bitmaps. These bitmaps indicate the total number of failures per cell type per wafer or per lot by the colour of the cells (darker colour indicates more failures). These bitmaps could reveal spatial effects. For an example see Figure 22.

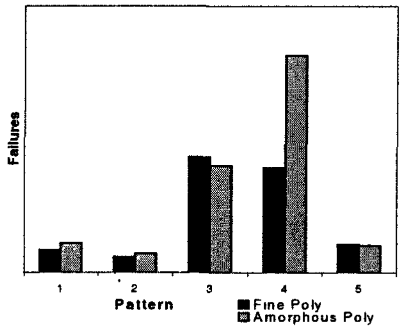


- Figure 22 Bitmap with a failing cell (left) and a stacked bitmap (right)

5.5.3 Pattern Failure Information

Besides determining which cell fails, it is also possible to exactly determine on which test pattern(s) the cell fails. This might be useful in determining where the problem exactly lies within a cell. However, no general method to do this is yet available, so it must be determined on a cell by cell basis.

The information can be obtained for each failing sub-module separately and from this, it is also possible to calculate totals for each pattern. This can for example be used in a comparative manner in the case of a split lot (a lot were some wafers have received a slightly different treatment than other). In Figure 23 an example can be seen. The graph shows the number of failures per pattern for a flip-flop (fd2) for the two halves of a split. It can be clearly seen that Pattern 4 fails relatively much more often for Amorphous Poly than for Fine Poly. As can be seen in Table 6, Pattern 4 checks whether the flip-flop is able to hold a value of 1 that was loaded earlier on. The cause of this problem is still under investigation.



- Figure 23 Comparison of two sides of a split on lot CZ8422 for flip-flop fd2

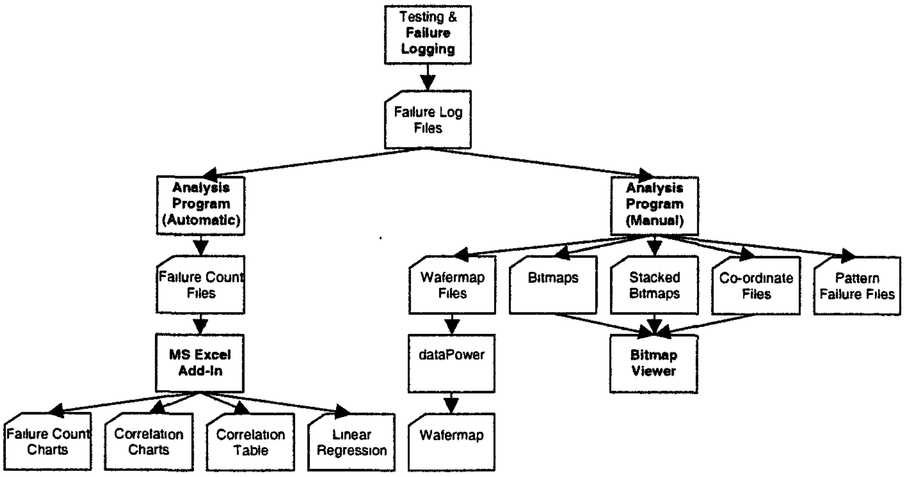
- Table 6 Test patterns of the fd2 flip-flop

pattern	function	clock	clear	D	Q	Q*
1	check load 0	1	1	0	0	1
2	check hold 0	0	1	1	0	1
3	check load 1	1	1	1	1	0
4	check hold 1	0	1	0	1	0
5	check clear	0	0	1	0	1

5.6 Tools

The deliverables for the first part of the project include the tools to support the diagnosis methodology. Developing these tools has taken a considerable amount of the total time spent on the project. This sub-section provides a short description of the functionality of the tools. It is not intended as a user guide neither as a detailed description of the internals of the software. For the first refer to [Lit. 4], for the second refer to the actual code and the comments included in it.

Figure 24 contains a graphic representation of the flow of data through the different programs. The rectangles represent tools, the nicked rectangles represent intermediate files. The programs that were specially implemented for the analysis of the RMDC modules are indicated with boldface.



- Figure 24 Information flow and analysis programs

The analysis trajectory starts with the logging of the failures on the tester. The log files that are generated are stored to a drive connected to its workstation. As soon as the testing of a wafer is completed, a script moves the log file from this workstation to a Unix server.

On this server, the incoming log files are processed automatically by a special analysis program. The numbers of failures per cell type per module type are calculated and these are stored in a text file that can be read by a spreadsheet program. This file and the original log file are moved to a directory with the lot number as name. Besides the totals per wafer, the totals per lot are calculated; these are also stored in the directory with the lot number as name.

To be able to trace back the cause of failures this analysis program needs a lot of input information. It includes the netlist information and layout information (for the bitmaps). The program actually reads in the standard layout files as produced by the design software.

If other information than the *failure counts* is required, this can be obtained by manually running the same analysis program. In this way bitmaps, wafer map files and extended pattern information can be produced from the original log file.

The bitmaps together with co-ordinate files are directly generated by the analysis tool. A special program for viewing the bitmaps, which runs under MS Windows, was also developed. This tool makes it possible to retrieve the co-ordinates of a failing cell by clicking on it.

The wafermaps are not directly generated by the analysis program; instead it creates a file that can be read with dataPower and dataPower can then be used to generate the actual wafermaps. dataPower is a program that is used in all Philips' foundries to analyse manufacturing data.

The pattern information is generated on a per sub-module basis. No programs are yet implemented for generating the statistical data. For the example in the previous sub-section, a custom script with a combination of Unix commands was used.

The actual diagnosis methodology is supported by an add-in for Microsoft Excel. This add-in can be directed to read in the file with the failure counts for the lot or wafer required. Then the add-in can produce a number of different types of chart, like the failure count chart and the correlation chart of Figure 18 and Figure 19 respectively. The add-in also includes the functionality of the main diagnosis method: the linear regression and automatic selection of the design parameters with the failure counts. The values of the extracted design parameters are included in the add-in. An example of the output of this analysis can be seen in Figure 20.

5.7 Conclusions

The methodology described in this section and especially the cell-level regression technique provides a powerful instrument for relating electrical failures in the RDMC modules to problematic process steps by statistical means. It also provides the information necessary as starting-point for failure analysis.

The different variants did not prove as useful as might be expected. The yield differences between them are only small and it is not possible to relate them to a single layer, but only to a group of layers. Moreover, yield differences can also be caused by other factors (like different orientation or power supplies for example).

A more useful method that is based on the differences in failure counts of the different cells was developed. By correlating the failure counts with the design attributes of the cells it is possible to determine which design attributes are involved in any excessive yield losses. This gives information that is much more detailed and it can also give an indication of problems in the cells themselves not only of problems with the routing.

This method cannot distinguish between design attributes that are closely correlated themselves however. It is of course also important that all the relevant parameters are extracted, although the method does give a clear indication if there are no attributes that can explain the pattern of failure counts.

The method was verified by comparing the results of the method with the results of special test structures on several lots. A strong correspondence was found and therefore it can be safely concluded that the method gives valid results and can at least identify the major causes of yield losses.

During this verification, the results of almost all considered lots indicated that a design attribute that was important for the yield losses was missing. This shows the power of the method for finding problems that are not targeted with special test structures. Still, the method does not make it possible to eliminate the special test structures because it does not give quantitative results as the test structures do.

If the results from this main diagnosis method show that further investigations are required, a number of additional analysis instruments are available. These are wafermaps, bitmaps and extended pattern information. However, no standard way to use them has been developed.

A lot of time was consumed by implementing the tools necessary to support the methodology. A combination of programs and scripts running on a Unix machine and a program and an Excel add-in running under MS Windows are used.

The methodology developed is both powerful and easy to use. It gives a good qualitative indication of design attributes that are causing yield losses. It provides a major advantage over using test structures, because it can detect and identify problems that are not targeted with the test structures.

6 Recommendations for Test Methods

To extract even more information from the RDMC modules it could be useful to employ additional test methods. The currently used stuck-at test method does not cover all possible (or even the important) failure mechanisms and is unable to distinguish between the fault mechanisms that are covered.

Besides all other criteria to which a good test method should adhere (like effectiveness in detecting a certain failure mechanism), the method should preferably also yield at least the level of detail that can be obtained with the stuck-at test: it should be able to pinpoint a *specific* failing cell. This makes it possible to use all the statistical methods described in Section 5.

There are three categories of test methods worth considering:

- Additional static patterns
- Current-based methods
- Timing-based methods

Each of these methods is considered in the following sub-sections. No experiments with the additional test methods were executed because of a combination of time constraints and questionable merits of the methods.

6.1 Additional Static Patterns

The patterns employed at the moment extensively test each cell separately for its static function. Therefore, all possible stuck-at faults and bridges between nets of the same cell are covered by design. The circuit is not specifically tested for bridges between nets connected to different cells. The coverage for bridging defects is therefore not complete.

In the original test set, which tested each cell after another, the coverage for bridges is higher because the surroundings of a cell are the same during all the testing of that cell. Therefore, all adjacent nets at least have an opposite value once. However, the coverage is far from complete because only one of the two nets is observed (and this net might dominate the outcome) and only one of the two combinations (zero on one net and one on the other) is tested.

The coverage for bridging faults could be improved by the addition of special patterns. However, the generation of these patterns is not trivial. It has to be guaranteed that all nets that are adjacent to each other for a certain length (that is have a large critical area) are tested with both combinations of opposite values. Testing means also that the outputs of both nets have to be observed.

A fairly straightforward solution would be to use the original, large test set, add observation of all nets and test all cells a second time, but now with the surrounding in the opposite state. This would of course *double* the original test time, which was already too large.

A better solution might be to modify the custom generation tool to add this functionality. Yet, this is complicated. In fact, you would need to implement large parts of the functionality of Amsal. (Amsal can generate patterns that target bridging faults based on the output of Interbridge [Lit. 10], which indicates the nets with large critical areas.)

The size of the pattern set is still likely to be a problem. Because most variants of the RDMC were routed to increase the critical areas, many nets will be adjacent to each other. To test all these combinations just requires a certain minimum number of patterns.

It still has to be seen, if the larger set of patterns would be very effective. The yield differences between the original, large test set and the currently used test set are very small as could be seen in Section 4.2.2. The large set is of course not yet optimal for the detection of bridging faults and it could be that two opposing effects cancel each other out (see also 4.2.2). It might thus be possible that the larger set could be (made) effective.

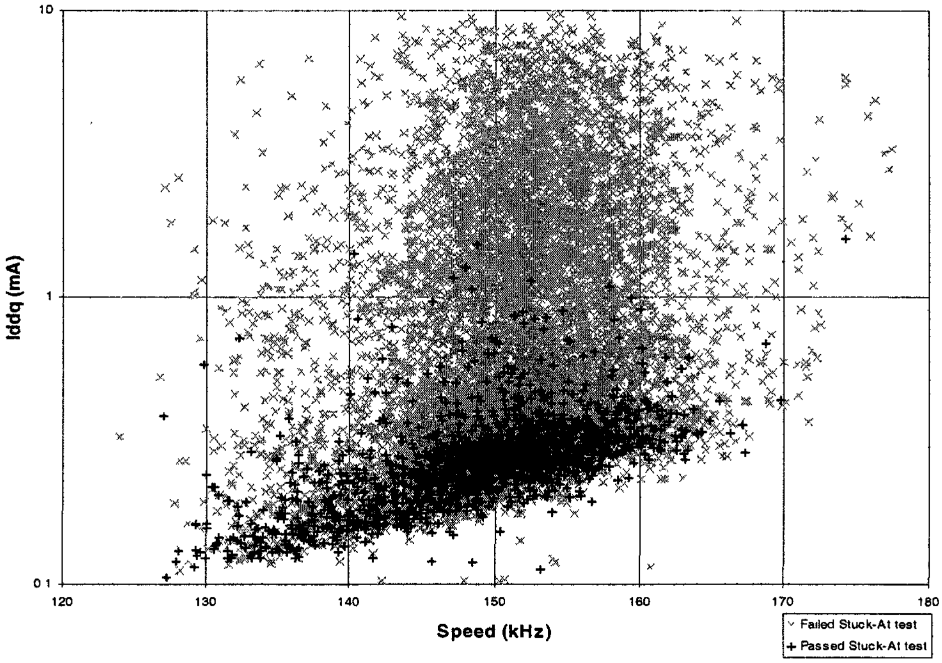
By using a larger set of patterns, the localisation potentially can be made more precise. It should be possible to distinguish between shorts to ground or vdd and bridges and even indicate which nets are involved in a bridge. This would require an advanced pattern generation strategy and additional analysis software, however.

6.2 Current-Based Methods

Current-based testing is based on the fact that a good CMOS circuit only has a very low quiescent current (the current that is drawn in the period between two clock edges). If the current is too high then some sort of problem is present in the circuit; it could either be a bridging defect or an open that causes a floating gate. This method is commonly known as Iddq-testing, which stands for Idd (the symbol for the current through the supply pins) Quiescent (while the circuit is in a steady state).

Implementations used for actual products measure the currents after setting up a device with special test vectors. Only a couple of vectors are used because of the time consuming nature of these measurements. Most often, a single limit on the current is set. If the current is higher than this limit the device fails, otherwise it passes.

A very basic Iddq-test is actually already implemented for the RDMC modules: they are initialised with all zeroes and then the current for all 12 RDMC modules together is measured. The measurement results are logged, but not used as pass/fail criterion. An example of the results (plotted against device speed) can be seen in Figure 25. It can be seen that some devices that were passing the stuck-at test would fail the Iddq-test if the limits would be set appropriately tight.



- Figure 25 Quiescent current (for all 12 RDMC modules together) versus the speed

A problem with current-based testing in deep submicron processes is that the leakage of good device is so high that the extra current due to a defect is obscured. Several improved current-based techniques are available like Transient Idd testing (measuring the current during a clock transition) and,Delta Iddq testing (using the difference between measurements on one device); the methods are called Iddt and Δ-Iddq testing respectively. In [Lit. 11] a comparison of the methods and recommendations for their usage are given.

For CMOS18, it is recommended that a form of Δ-Iddq testing is used where the currents for different patterns are compared to each other or to a golden reference (called current signature). In the case of the RDMC modules, the signature-based approach will probably be necessary because large differences in current between the patterns are expected. Experiments will have to confirm this however.

Because the Veqtor is used in the early stages of process introduction, the currents can shift quite dramatically and can have a large spread (as can be seen in Figure 25); also, there are many faulty devices. Therefore, it will be difficult to establish a usable signature and compare the circuits against this signature.

The Δ-Iddq method requires a different current limit for each test pattern. For a flexible implementation, it is necessary to measure the currents instead of comparing it against a limit. However, this takes a lot of test time; especially when a large set of test vectors is used. This might be a severe limit to the application of this test method.

To be able to use the current-based tests together with the diagnosis methodology discussed in Section 5, it is necessary that more detailed information can be obtained than only a pass or fail of all the RDMC modules together. The least would be to be able to determine a yield per RDMC module; preferably failing cells should be indicated.

Even though the current of all RDMC modules together is measured, it is possible to determine the failing modules, because they can be tested separately. However, no distinction can be made between the 10 RDMC sub-modules because the data is shifted in from the same input in parallel. Fortunately, this is not a very large disadvantage for the statistics (although faults in multiple sub-modules might not be detected as such). For failure analysis, it might be a larger problem (although the search is narrowed down to 10 locations and it might be quite easy to find the problematic sub-module by using liquid crystal to detect a hot spot).

To be able to determine the specific cell that most likely cause the increase in current, special patterns need to be generated. The straightforward approach would be to test each cell after another. If there is an increase in current for a specific pattern, this can be directly related back to a cell. This would mean a large number of patterns however and the correspondingly large number of current measurements (and thus a large test time).

The number of patterns needed might be reduced considerably by using a binary search if only a single defect is considered. (See the example of Figure 26.) Each cell is tested $\lceil {}^{2}\log(\# \text{ of cells}) \rceil$ times, but the number of consecutive runs can be reduced to $2 \times \lceil {}^{2}\log(\# \text{ of cells}) \rceil$. This would mean that instead of 3440 patterns only $2 \times \lceil {}^{2}(\log(297)) \rceil \times 64 = 1152$ patterns (where 64 is the maximum number of

C1	C2	C3	C4	C5	C6	C7	C8	Test Result
X	X	X	X					F
				X	X	X	X	P
X	X			X	X			F
		X	X			X	X	P
X		X		X		X		P
	X		X		X		X	F

X means that the cell is tested during that run. The gray boxes indicate the ısolation process

- Figure 26 Example of the binary search technique patterns per cell) would be needed. If the few cells with more than 16 patterns would be tested consecutively, this could be reduced to $2 \times 32 + 6 \times 64 + 2 \times \lceil {}^{2}(\log(289)) \rceil \times 8 = 592$ patterns.

The problem with this test scheme is of course again that no standard tools are available to generate these patterns. Either the original (large) pattern set could be used or the custom program has to be heavily modified.

6.3 Timing-based Methods

A number of timing-based test methods exists. All these methods rely on the fact that defects (like resistive opens and bridges) introduce a delay at a certain point in the circuit. The simplest form is running normal stuck-at patterns at higher speeds. Better methods use specially generated patterns that try to toggle the value at each node (this is the only situation in which a delay can be detected).

To be effective the cycle length has to be only slightly above the normal delay of the critical path. For normal random logic, which has a relatively long critical path (at least 20 gates), it is not such a problem. However, for the RDMC modules, with a logic depth of only one, the clock period would have to be in the order of a nanosecond.

The tester does not support cycles shorter than a couple of nanoseconds and it would probably be impossible to get the signals to the device via the pads anyway. The required clock frequencies could thus only be generated onboard. Because the circuitry for this is not in place, it is impossible to execute these kinds of tests on the RDMC modules at the moment.

6.4 Conclusions

Generating additional test patterns can be interesting because of the potentially more precise localisation of bridges. However, it is a major effort to generate suitable patterns and it also will require some large changes to the analysis software.

The current-based methods are interesting, because they are able to detect extra categories of defects, but investigation into the feasibility is necessary. It is uncertain whether the signature-based approach can be used successfully (especially in the early phases of process introduction). Also in this case, the pattern generation will be difficult and the analysis software will have to be changed.

No timing-based tests can be executed on the current design of the RDMC modules because of the high clock frequencies that would need to be generated by the tester and fed into the device.

Not enough time remained to do any experiments with either extra patterns or current-based testing. The current-based approach seems to be the most promising because it covers additional failure mechanisms, but it will also require the most time to implement.

7 Recommendations for Changes to the Design

The RDMC modules are working properly and a useful diagnosis methodology can be based on the generated data. Changes to the design are therefore not strictly necessary. However, during the work with the modules a number of points have come up that could be improved.

Before the merits of any changes can be evaluated, the criteria for the design have to be clear. The purpose of the modules is to have reasonable representative logic that is easily analysable. The exact balance between these two is of course difficult to define. It has to lie somewhere in between the RDMR modules (which contain a DSP and are thus representative) and special test structures (which are easily analysable).

The following points that could be improved have been found:

- The module is not well suited for detecting opens: this is due to the impossibility of implementing timing-based tests and to the difficulty of implementing a good current-based test.
- The area overhead of the scan flip-flops is high: this makes the module less representative and also causes a relatively high loss on scan-continuity testing.
- The routing of the modules is not particularly dense: the modules are not as sensitive as possible to problems in the layers they were intended to target.
- The area of a single RDMC sub-module is rather small: the overhead for power routing and empty areas in a complete RDMC module is relatively high.
- Similar cells are grouped together geographically and every cell is placed only once: this makes it more difficult to determine whether a problem is caused by the design of the cell or that effects have to do with the location.
- The inputs of all the sub-modules are connected together: this makes it impossible to test them separately, which would be valuable for Iddq-testing.
- The different routing variants are only of limited use: this is due to the small number of different variants.

The next sub-sections will go into these problems one by one (except for point 4 and 5, which are related and will be discussed together in one sub-section).

7.1 Detection Capability for Opens

The preferable method to implement a detection capability for opens would be to make timing-based tests possible. However, this is problematic because there is only one layer of cells and thus the required clock period is very short (about a single nanosecond).

Thus, the basic problem here is the single layer of cells. However, adding more layers is not possible if the circuit is to be kept easily analysable: not all cell types can be chained in a useful and controllable manner. If cells of different types would be used together, it would be problematic to assign a failure to a certain type of cell.

It is of course possible to create the required cycle times by means of an on board clock generator. However, real Phase Locked Loop circuits were not available at the moment the Veqtor was designed (and this is not likely to be the case for a Veqtor for a new process generation). Another solution might be to use a delay line (which can be tapped at a programmable point) to generate the required clock signals. However, it is difficult, if not impossible to obtain absolute timing measurements with delay lines.

In addition, test patterns that cause nodes to toggle are needed. (Otherwise timing-based methods are of no use) This will require extensive modifications to the test generation program. However, in this case no changes to the analysis software will be required at all, because it still can be determined which scan-chain flip-flop contains a faulty value and relate this directly back to a failing standard library cell.

Even though a capability could be added to execute timing-based tests to the RDMC modules, it is uncertain if it is of much use. The circuit would still not be very representative for normal random logic for timing related issues (for example because there would still only be a load of one cell connected to each cell). It is therefore questionable if effort should go into these kinds of developments. It might be better to concentrate on implementing timing-based tests for other modules on the Veqtor.

7.2 Area overhead of Flip-Flops

The ratio of flip-flops to other cells is rather high for the RDMC modules. To be more representative, it would be better to reduce the area occupied by this overhead. However, all in- and outputs of the standard library cells have to stay directly accessible because of the analysability. For the inputs, this is no problem, instead of connecting them to the outputs of flip-flops they could be directly connected in parallel to a number of inputs. However, the number of scan-chain flip-flops is determined by the number of outputs of the standard library cells.

What is achieved by the flip-flops is that the 319 outputs of the library cells are multiplexed to a single output. The same functionality is possible with a large multiplexer instead of the flip-flops. In this circuit the inputs of the library cells would be connected to the global inputs in parallel and the multiplexer would serve as a large switch to select a single output.

The design with multiplexers has certain advantages:

- The area occupied by multiplexers is smaller than the area occupied by flip-flops:

  For N outputs of the library and using an x-input multiplexer the number of necessary layers L of multiplexers can be expressed as:

  $$L = \left\lceil {}^{x}\log(N) \right\rceil$$

  Each layer contains a factor x less multiplexers. Therefore, the number of flip-flops F can be expressed as the sum of the flip-flops in all layers:

  $$F = \sum_{i=1}^{L} \left\lceil \frac{N}{x^{i}} \right\rceil = \sum_{i=1}^{\left\lceil {}^{x}\log(N) \right\rceil} \left\lceil \frac{N}{x^{i}} \right\rceil$$

  If 2-input multiplexers are used 321 multiplexers are necessary for 319 inputs. As these multiplexers occupy only about 60% of the area of the flip-flops, a serious area reduction is achieved. If 4-input multiplexers are used an even larger reduction is achieved because only 108 multiplexers are necessary and these multiplexers are about 133% of the size of a flip-flop. The total area for the 4-input multiplexers therefore is only be about 45% of the area of the flip-flops.

- The test time could be reduced:

  In the design with the multiplexers every clock cycle can be used effectively, while in the design with the flip-flops some cycles are wasted by shifting in and out values that are not actually used.

- All input lines are directly controllable:

  This makes it possible to test the flip-flops more extensively for their data retention capabilities etc.

However, there are also some important disadvantages:

- The circuit would be less representative of normal logic because of the absence of scan-chains.
- No scan-continuity testing is possible. Thus, it is not possible to test if the auxiliary circuitry (the multiplexers in this case) is working properly. However, it is possible to add some kind of testability of the multiplexers by adding an extra layer of multiplexers and connecting one input of each of the first level multiplexers directly to the global inputs. This would mean 622 2-input multiplexers (which would occupy 122% of the area of the flip-flops) or 144 4-input multiplexers (60% of the area).
- The number of inputs needed is increased: instead of 1 scan-chain input and 4 global nets, 9 (for 4-input multiplexers) or 10 (in the case of 2-input multiplexers) selection lines and 8 global nets are necessary. Although this fits into the Veqtor concept, it will not be possible to test the sub-modules separately as suggested by the design change of Section 7.5.
- Timing-based testing is not useful because the time that would be measured would be largely determined by the delay time of the multiplexer tree.
- All software tools surrounding the RDMC module (for design, testing and diagnosing) would have to be changed.

The advantages of the design with multiplexers do not seem to outweigh the disadvantages. It is possible to use a combination of the two options, but this complicates matters, without having any benefits over the two extremes. The original goal to make the circuit more representative of normal logic cannot be achieved in this way.

7.3 Denser Routing

The routing of the RDMC modules is not particularly dense neither in an absolute sense nor when comparing it to for example the RDMR modules. This makes that the modules are not as sensitive as possible to problems in the metal layers as possible.

To achieve a denser routing the flip-flops were rotated 2 places (as explained in Section 3.3). According to the design documentation, it became impossible to route the modules in 2 layers if this distance was increased. Anyway, the rotating causes a peculiar routing pattern, where a couple of nets are long (the ones connected to the flip-flops that were moved from the end to the beginning of a row) and all the others are short.

An alternative for the rotation is to randomly change the places of the flip-flops. However, this should be steered so that the flip-flops are not moved to far from their original location. An Excel sheet was made to accomplish this. The optimal distance is different for the different variants of the RDMC because for metal 5 and 6 the routing pitch is double that of the other metal layers. To find the optimal distance different placements were actually routed. In the end it turned out that an average distances of 6.2 was achievable for a variant in metal 2 and 3 and a distance of 4.0 for a variant in metal 4 and 5.

The routing for these new variants is almost maximally dense. A comparison between the old and the new versions in terms of wire lengths and via counts can be found in Table 7. The critical area for small defects is even increased with a factor of about five. An example of the old and the new routing can be seen in Figure 27.

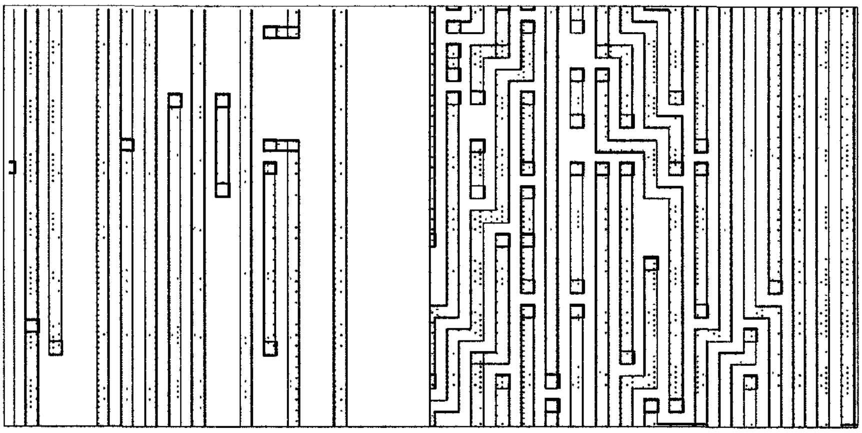
- Table 7 Routing Characteristics of the new RDMC's

Length	rdm2c	rdm5c	rdm3c	rdm6c
metal2	20.5	56.4	2.2	2.6
metal3	26.2	57.3	2.6	4.3
metal4	0.4	6.2	23.9	48.1
metal5	-	-	20.1	30.3
via1	3059	3049	3030	3009
via2	2212	9399	3013	2950
via3	74	2619	3792	3939
via4	-	-	2728	4995

These new variants are not yet completely finished:

- No timing analysis for clock skew has been performed. It is not expected that this will give problems because the blocks are very small. To make sure the timing analysis has to be done however.
- Only the sub-modules are finished, they are not yet combined into complete RDMC modules and they have not yet been checked for design rule violations (DRC) or for discrepancies with the netlist (LVS).

The denser routing has been achieved with a relatively low amount of work. It is recommended to use this strategy for all new variants of the RDMC because the modules will become much more sensitive to defects.



- Figure 27 Part of the routing of the old rdm2c on the left and the new rdm5c on the right

7.4 Area Considerations and Multiple Instances of the Cells

The area of the separate RDMC sub-modules is rather small. It makes the circuit less representative for normal random logic and it causes a relatively large overhead for the power routing etc. Another problem is that every cell type is placed only once and that cells of similar types are grouped together. This makes it difficult to determine if problems are caused by the location or the design of the cell.

It is easy to solve all these problems by placing multiple instances of each cell type in an RDMC module and not grouping cells with similar functionality together. This has as disadvantage that the physical localisation of defects in the back-end will become more difficult because length of the nets is increased.

It has to be considered whether all cells will be placed together in one large scan-chain or whether they will be distributed over multiple scan-chains. Distributing them over multiple scan-chains reduces the necessary test-time, but the degree of distribution is limited by the number of output pins available (16 according to

[Lit. 1]). In addition, it requires more change to the diagnosis software because the different chains have to be treated as different module types.

The optimum number of sub-modules that is combined into a single RDMC module will depend on how many instances of each cell are placed and what size is desirable for the RDMC modules. This has to be considered afresh for each new process generation. Still placing at least 2 instances of each cell seems desirable. Placing more is only beneficial for the area overhead, not for the diagnosability aspect.

7.5 Separate Testability of the Sub-modules

For localising defects by using Iddq-testing, it is desirable to be able to test the sub-modules separately. This can be achieved by using separate inputs for the RDMC sub-modules instead of connecting them together. It is not really necessary to duplicate the global inputs for every net (although this would be preferable); separating the scan-chain inputs would be enough. This solution poses no problem with the structure of the Veqtor, as a maximum of 25 input pins is allowed.

7.6 Optimal Variants

The current number of different variants does not make it possible to pinpoint a specific failing metal or via layer. To make this possible the number of variants would have to be much larger. This is not desirable because this would require too much area (every type has to occupy a certain area to be able to detect small yield losses). Four different types is probably about the maximum.

It can be questioned if the variants are useful at all if they cannot pinpoint specific layers. Of course, they do give an indication, but this is also given by other test structures and the more detailed diagnosis methodology proposed. However on a couple of lots the RDMC modules picked up an unexpected signal that was not seen on any other structure. Besides, it might be problematic to develop an RDMC that uses all routing layers and also has dense routing in all the layers. Furthermore, the different variants do not cost overmuch time to develop, so it is still worthwhile to include several different variants.

It can also be questioned if the current set of variants is optimal or if other variants might be more useful. The variant with a minimum of routing is a good reference, so this should be retained.

To make pinpointing a problem as easy as possible the number of layers in the other variants should be as small as possible. It is impossible to route a module in less than two layers, so the choice to route in two layers seems justified. These layers should be directly on top of each other to minimise the number of vias between other layers.

The variants with routing in metal layers 2 and 3 respectively 4 and 5 seem to be good choices. The variant with routing in metal 2 and 6 to maximise the number of vias did not work out very well however (see Section 3.3). The use of a variant with many vias in all layers is questionable anyway, because it would be impossible to distinguish between problems in the different via layers. It might be better to produce a variant in metal 5 and 6. This would add a good coverage for via 5 and improve the density of metal 6.

7.7 Conclusions

Although all the problems with the RDMC modules in theory could be solved as was seen in the previous sub-sections, some require changes that make the module less suitable in other respects. In addition, some of the changes might require quit a lot of time to implement (also because of the need for changes to the surrounding software).

Changing the design to allow for timing-based tests (to add better detection capability for opens) is possible to a certain extent. However, the results that can be obtained do not seem to be worth the effort and it is probably better to concentrate on other blocks for timing-based tests.

Although reducing it is possible to reduce the area occupied by auxiliary circuitry by using multiplexers instead of flip-flops, this does not make the circuit more representative of normal random logic, but even less. It does have some other advantages, but these do not outweigh the disadvantages.

The following changes to the circuit are useful and can be implemented without compromising other aspects:

- Denser routing of the modules by randomly swapping the positions of the flip-flops.
- Placing multiple instances of the standard library cells in each sub-module (this will also increase the area of the separate sub-modules) and not grouping cells of a similar type together.
- Separating the scan-chain inputs of the sub-modules.
- Replace the variant that is routed in metal 2 and 6 (rdm4c) with a variant with routing in metal 5 and 6 layers.

None of these changes affect the basic ideas behind the RDMC modules. They are only fairly minor changes in the details of the implementation and should not require overmuch effort. All can be used with no or relatively small changes to the existing software.

8 Conclusions

8.1 Test Patterns and Failure Logging

Before the actual work on the development of an optimal diagnosis methodology could be started, a new set of test patterns for the RDMC modules had to be implemented. The original set just required too much test time to be usable.

A new set was generated with the help of a custom tool that was a factor 60 smaller than the original one. However, when the new patterns were used, it was discovered that the yield with the new set was slightly higher than with the old set and thus the coverage was lower.

Several possible causes were determined and other pattern sets were generated to establish the real cause. It turned out that the interval between shift out and shift in of the data was the main factor causing the yield differences. In the first version of the new pattern set, this interval was removed. After including the interval again, the yield was about the same as with to the original set. The set is still about 30 times smaller than the original one after the interval was added again.

The yield differences between the different pattern sets were examined in more detail. It turned out that the differences were caused by the flip-flops included in the RDMC modules. They fail more often when the interval between loading the data and checking if it is still retained is longer. This problem might also occur in normal random logic and further studies into this are recommended.

For a detailed analysis of the RDMC modules, it is necessary to log which test patterns fail for the failing devices. By using several techniques to reduce the amount of data, it proved possible to make the logging fast enough to log each and every lot that is tested. This is very valuable because in this way all the silicon that is produced can be used for the detailed analysis.

8.2 Diagnosis Methodology

With all the preliminaries in place the work on the diagnosis methodology was started. The methodology consists of a number of different ways to look at the RDMC modules. The methodology relies heavily on the special features of the RDMC modules of course.

The yield differences between the different variants of the modules can give a first indication of problems on the metal or via layers. However, nothing more than a rough indication of the problems can be obtained due to several causes: the low sensitivity of the modules for defects; other possible causes of yield differences (like different power supplies and orientation); and because they only can give an indication of a set of metal and via layers that are problematic.

The method of correlating the failure counts per cell with the design attributes is far more successful and gives an indication of the attributes that are problematic. These can be attributes of the cells themselves as well as attributes of the routing connecting the cells. The method is very easy to apply because of the tools supporting it and directly gives a list of design attributes that are most likely to be involved in excessive yield losses.

The method has problems with distinguishing design attributes that are closely correlated (and there are quite a lot of these attributes). It is also important to have the correct parameters extracted although it can be determined from the output if there is an important explaining attribute missing.

It was verified if the method gives valid results by comparing it against the results from special test structures. Because the correspondence is quite good, it can be concluded that the method gives valid results and can at least indicate the main causes of yield loss.

Of course, most of the problems that are detected with the RDMC modules could also be detected with special test structures, but —and this is very important— not all. Then again, the RDMC modules cannot remove the need for test structures because it does not give absolute quantitative information, but only qualitative information.

If the results of this main diagnosis method require further investigation a number of additional analysis instruments are available. These are wafermaps, bitmaps and information about the exact patterns that fail. However, no standard way to use them has been developed

The diagnosis methodology has already proven to be useful. It has identified some process problems that could never have been traced back to the exact cause without the methodology. Because its ease of use, it does not have to take much time to monitor all lots and this can provide very valuable information.

8.3 Changes and Additions to the Test Methods and the Design

After the diagnosis methodology was developed, the attention was turned to changes in or additions to the test methods employed. Considered were the generation of extra test patterns to detect bridging faults better and the use of current-based and timing-based test methods to detect additional types of defects.

It is possible to be very short about the timing-based methods. It is impossible to use these with the current design of the RDMC modules because of the very high clock speeds that would have to be generated outside of the device and fed into it.

The generation of additional test patterns that are better able to detect and isolate bridging faults can be interesting. However, it requires extensive changes to both the pattern generation program and the analysis software.

The use of advanced current-based methods is interesting because they cover additional categories of defects. If the current-based methods are implemented in the correct way it is still possible to use all the diagnosis methods that were described earlier on (with some changes to the analysis software). It is uncertain whether the current-based methods can be used successfully (especially in the early phases of process introduction). Another problem is again the generation of the required test patterns.

So although additional test methods might be interesting, they will require quite some time to implement and the results are uncertain. However, it might be necessary to spend this effort when failure mechanism that are not covered by stuck-at testing become predominant in current or future processes.

Finally, alterations to the design of the RDMC modules were considered. Most of them address issues that were found during the use of the RDMC module. All the issues could be tackled in some way or another, but some changes would make the module less suitable in other respects or might require large amounts of time to implement.

Changing the design to allow for timing-based tests is possible by generating the required frequencies on board by means of for example a delay line. However, the circuit would not be very representative for normal logic with respect to timing because of the logical depth of 1. Furthermore, it would be necessary to generate special test patterns that cause nodes to toggle and this would require some large changes to the pattern generation software.

The area occupied by scan flip-flops is relatively much higher than for normal random logic. It is possible to reduce the number of scan-chain flip-flops (even to 0) by using multiplexers instead. However, this does not make the circuit more representative of normal random logic, but even less. It does have some other advantages, but these do not outweigh the disadvantages.

All the other changes that were considered are feasible and do not require excessive amounts of time. Implementing these changes will make the RDMC even more useful. The following changes are suggested:

- Denser routing of the modules by randomly swapping the positions of the flip-flops.
- Placing multiple instances of the standard library cells in each sub-module and not grouping cells of a similar type together.
- Separating the scan-chain inputs of the sub-modules.
- Replace the variant that is routed in metal 2 and 6 (rdm4c) with a variant with routing in metal 5 and 6.

No major changes to the design of the RDMC modules are suggested, this reflects the notion that the current implementation of the RDMC modules is already very useable. The changes that are suggested only improve upon the current design.

8.4 Overall Conclusions

The developed diagnosis methodology is both powerful and easy to use. It gives a good qualitative idea about the process steps that are causing difficulties. A major advantage (compared to test structures) is it is able to identify unexpected problems, which are not targeted with test structures.

As the usability of the developed methodology clearly shows, the current tests and design of the RDMC module are sufficient. However, additions to the test methods could be interesting and even might become necessary although they will require substantial effort. No major changes to the design of the RDMC modules are recommended, but some relatively small changes could improve the diagnostical capabilities of the modules.

All in all it can be safely stated that the RDMC modules together with the developed diagnosis methodology are a powerful tool for yield ramping. Some minor changes to the design of the modules could even further enhance this capability.

Acknowledgements

I wish to thank many people for their time, effort, knowledge they shared with me or just being there as colleges. I wish to mention the following people by name however.

The people from the DfM section: First of all Paul Simon, for sharing his knowledge about yield models, test structures and DfM in general and for his input for the project and the reviewing of this report. Marc van de Pol, for answering many questions about the design and extraction tools. Paul Volf, with whom I worked closely together on the testing of the Veqtor. Martin Visser for his help with Cadence and Sedsm in particular. Bram Kruseman for his knowledge about current-based testing. Diederik Fokkema for his knowledge about mathematics and some very interesting phrases. The other people of the DfM section should not go unmentioned and although their contributions are less concrete they are valued no less: Eric van de Wetering, Jose Pineda de Gyvez and Tony Jurg.

Then there are a number of other people from Philips that I would like to thank: Dirk de Vries, Tony Gummer, Walter de Coster, Nebosja Jankovic, Guidi Fleuren and Rene Wientjes.

I also would like to thank Rene Segers and Kees Veelenturf for their work as respectively supervisor and coach; for keeping an eye on my progress, their reviewing of this report and of course making this project possible in the first place. I also would like to thank prof. Stevens for acting as the third member of the graduation committee and second examiner for this graduation project.

Literature

[Lit. 1] Wiel, Paul van de.
Veqtor4 Specification.
Nijmegen: Philips Semiconductors B.V., MOS4YOU, (I:\Groups\Procdev.ms4\DFM\ProductDocuments\Veqtor4_CMOS18\Veqtor4spec.doc), 2000.

[Lit. 2] Wiel, Paul van de.
Veqtor4 RDM, Design manual for the random logic modules.
Nijmegen: Philips Semiconductors B.V., MOS4YOU, (I:\Groups\Procdev.ms4\DFM\ProductDocuments\Veqtor4_CMOS18\Veqtor4rdm.doc), 2000.

[Lit. 3] Schanstra, Ivo.
rdmc testpatroon generatie programma: genrdmcpat.
Nijmegen: Philips Semiconductors B.V., MOS4YOU, (I:\Groups\Procdev.ms4\DFM\OverdrachtVanIvoAanPaul\genrdmc.doc), 2000.

[Lit. 4] Mels, Arjan.
VEQTOR4 RDMC Tools.
Nijmegen: Philips Semiconductors B.V., MOS4YOU, (as yet unpublished).

[Lit. 5] Chang, Jonathan. T.-Y. et al.
Analysis of pattern-dependent and timing-dependent failures in an experimental test-chip.

[Lit. 6] Sze, S.M.
VLSI technology.
New York: McGraw-Hill Book Company, 2nd edition 1988.
McGraw-Hill Series in Electrical Engineering. Electronics and electronic circuits.

[Lit. 7] Ferris-Prabhu, A.V.
Role of defect size distribution in yield modelling.
IEEE Trans. Electron Devices, vol. ED 32, no. 9, pp. 1727-1734, 1985.

[Lit. 8] Anderson, H.P. and Blankespoor, J.H.
Inleiding tot de statistische analyse.Praktische statistiek voor het hoger beroepsonderwijs.
Rijswijk: Uitgeverij Nijgh & Van Ditmar, 2e druk 1990.

[Lit. 9] Chatterjee, S. and Price, B.
Regression Analysis by Example.
New York: John Wiley & Sons, 1977.
Wiley series in probability and mathematical statistics.

[Lit. 10] Interbridge Reference Manual Version 3.0.
Eindhoven: Philips Electronics N.V., CAT Support,
(http://pww.research.philips.com:25138/Test/CAT/manuals/interbridge/interbridge-1.html)

[Lit. 11] The Future of Current-Based Testing - Iddx Task Force Recommendations.
Eindhoven: Philips Semiconductors, IddX Task Force,
(http://pww.natlab.research.philips.com:25138/Test/QuicMon/iddx-tf/iddxtf_recom.pdf)

[Lit. 12] IDDQ threshold limit.
Eindhoven: Philips, IddX Task Force,
(http://pww.natlab.research.philips.com:25138/Test/QuicMon/calc/threshold.html)

Appendix A Extracted Design Attributes

A.1. Front-end and Metal 1 attributes

Only the parameters that are unequal to zero for at least one cell are summarised here.

Parameter Name	Explanation
CA_A_SALI_????	Critical area for active shorts
CA_A_POLY_????	Critical area for poly shorts
CA_A_ILIL_????	Critical area for LIL (Local Interconnect) shorts
CA_A_METAL1_????	Critical area for metal 1 shorts
CA_A_ILILSALI_????	Critical area for LIL to active shorts
CA_A_ILILPOLY_????	Critical area for LIL to poly shorts
CA_A_SALIPOLY_????	Critical area for active to poly shorts
CV_A_ACT	Area covered by active
CV_A_NWE	Area covered by Nwell
CV_A_POL	Area covered by poly
CV_A_PPL	Area covered by Pplus
CV_A_LIL	Area covered by LIL
CV_A_CNT	Area covered by contacts
CV_A_ME1	Area covered by metal 1
IC_NR_ACT_CNT	# of contacts to active
IC_NR_POL_CNT	# of contacts to poly
IC_A_ACT_CNT	Area of contacts to active
IC_A_POL_CNT	Area of contacts to poly
GO_NR_PMOST	# of pmos transistors
GO_NR_NMOST	# of nmos transistors
GO_NR_CORN_135	# of 135º corners in the gates
GO_NR_L_0.18_0.23	# of transistors with a length of 0.18-0.23µm
GO_NR_L_0.23_0.28	# of transistors with a length of 0.23-0.28µm
GO_NR_W_0.28_0.4	# of transistors with a width of 0.28-0.40µm
GO_NR_W_0.4_1	# of transistors with a width of 0.4-1µm
GO_NR_W_1_3	# of transistors with a width of 1-3µm
GO_NR_W_3_5	# of transistors with a width of 3-5µm
GO_A_PMOST	Area covered by the gate oxide of pmost's
GO_A_NMOST	Area covered by the gate oxide of nmost's
GO_A_LOCOS_EDGE	Length of the edge between poly and the field oxide
GO_A_POLY_EDGE	Length of the edge between poly and active

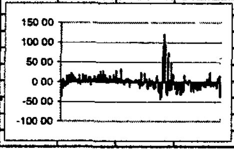
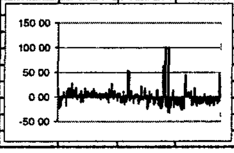
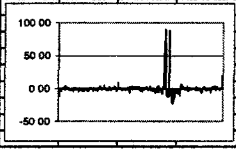
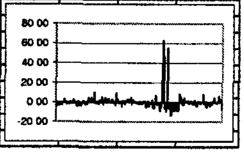
For the critical areas the question marks have to be filled in with: 0005, 0010, 0015, 0020, 0025, 0030, 0035, 0040, 0045, 0065, 0085, 0105, 0125, 0155, 0185, 0230, 0270, 0330, 0400, 0500, 0600, 0700, 0900, 1100, 1300, 1700, 2100, 2500 and 3000. The particle sizes in micrometers are the numbers divided by 100.

A.2. Metal 2-6 and Via attributes

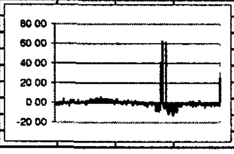
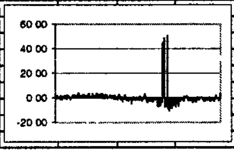
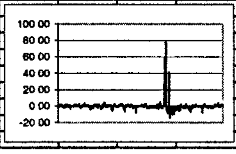
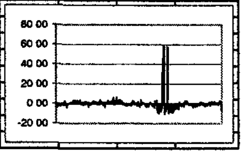
Parameter Name	Explanation
rdm?c M1-6	Length of metal 1 (the metal in the standard library cells is not included)
rdm?c M1-6	Length of metal in the connected nets
rdm?c? via 1-5	Number of vias in the connected nets

Appendix B Results of RDMC analysis

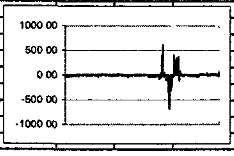
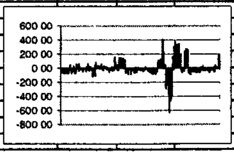
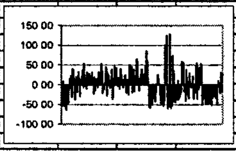
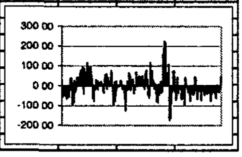
B.1. Lot CZ9700

	Total rdm1c 1800mV 100ns				Total rdm2c 1800mV 100ns				Total rdm3c 1800mV 100ns				Total rdm4c 1800mV 100ns			
R^2	0 33				0 16				0 19				0 16			
Significance	0 00				0 00				0 00				0 00			
F	70 74				26 86				33 38				27 23			
Critical F	3 03				3 03				3 03				3 03			
SEy	17.90				16 57				10 78				7 40			
	Beta	Sign	Coef	Conf Int	Beta	Sign	Coef	Conf Int	Beta	Sign	Coef	Conf Int	Beta	Sign	Coef	Conf Int
Const		0 00	32 20	4 64		0 00	29 70	4 29		0 00	7 16	2 79		0 00	12 01	1 92
CA_A_ILIL_0045	0 28	0 00	0 95	0 42	0 33	0 00	0 94	0 39	0 26	0 00	0 48	0.25	0 34	0 00	0 42	0 17
CA_A_METAL1_0015	0 35	0 00	4929 49	1730 46	0 09	0 22	996 93	1602 16	0.22	0.00	1705 84	1042 36	0 09	0 22	448 48	715 73
Residual Plots																

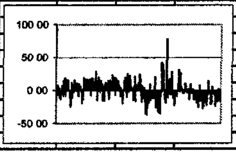
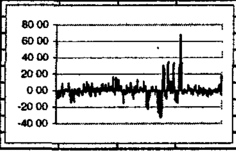
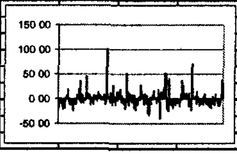
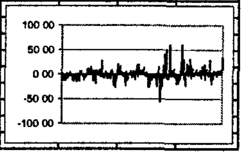
B.2. Lot CZ9701

	Total rdm1c 1800mV 100ns				Total rdm2c 1800mV 100ns				Total rdm3c 1800mV 100ns				Total rdm4c 1800mV 100ns			
R^2	0 49				0 28				0.37				0 17			
Significance	0 00				0 00				0 00				0 00			
F	137 94				57 12				83 44				29 43			
Critical F	3 03				3 03				3 03				3 03			
SEy	7 55				6 55				7 86				6 87			
	Beta	Sign	Coef	Conf Int	Beta	Sign	Coef	Conf Int	Beta	Sign	Coef	Conf Int	Beta	Sign	Coef	Conf Int
Const		0 00	7 96	2 11		0 00	4 87	1 83		0 00	3 50	2 03		0 00	3 01	1 90
M4	0 59	0 00	0 00	0 00	0 44	0 00	0 00	0 00	0 48	0 00	0 00	0 00	0 29	0 00	0 00	0 00
CV_A_LIL	0 16	0 00	0 57	0 38	0 13	0 04	0 34	0 33	0 17	0 01	0 57	0 40	0 16	0 02	0 42	0 35
Residual Plots																

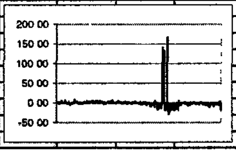
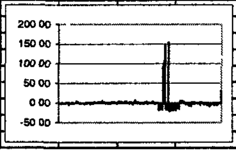
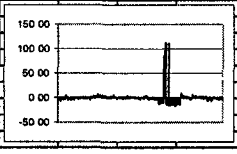
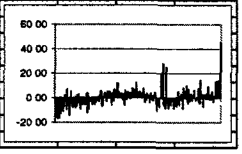
B.3. Lot CZ9893

	Total rdm1c 1800mV 100ns				Total rdm2c 1800mV 100ns				Total rdm3c 1800mV 100ns				Total rdm4c 1800mV 100ns			
R^2	0 83				0 76				0 36				0 84			
Significance	0 00				0 00				0 00				0 00			
F	727 48				453 20				82 46				743 18			
Critical F	3 03				3 03				3 03				3 03			
SEy	104 05				117 56				33 73				53 45			
	Beta	Sign	Coef	Conf Int	Beta	Sign	Coef	Conf Int	Beta	Sign	Coef	Conf Int	Beta	Sign	Coef	Conf. Int
Const		0.04	30 51	28 84		0 20	21 16	32 59		0 00	49 18	8 46		0 00	59 43	14 01
VIA3	0 87	0 00	25 52	1 79	0 73	0 00	22.47	2 29	0 32	0 00	0 66	0 26	0 74	0 00	3 75	0 32
CA_A_ILIL_0065	0 06	0 05	0 96	0 96	0 20	0 00	3 02	1 09	0 33	0 00	0 87	0 33	0.24	0 00	1 98	0 51
Residual Plots																

B.4. Lot CZ8422

	Total rdm1c 1800mV 100ns				Total rdm2c 1800mV 100ns				Total rdm3c 1800mV 100ns				Total rdm4c 1800mV 100ns			
R²	0 71				0 88				0 50				0 75			
Significance	0 00				0 00				0.00				0 00			
F	346.76				1064 36				143 22				425.50			
Critical F	3 03				3 03				3 03				3 03			
SEy	14 55				10.05				15 31				13 07			
	Beta	Sign	Coef	Conf Int	Beta	Sign	Coef	Conf Int	Beta	Sign	Coef	Conf Int	Beta	Sign	Coef	Conf Int
Const		0 00	36 73	4 36		0.00	24 68	3 01		0 00	35 63	4 92		0 00	40 56	3 94
VIA3	0 80	0 00	2 45	0 19	0 92	0 00	3 45	0 15	0 62	0 00	0 65	0 09	0 81	0 00	0 81	0 06
IC_NR_ACT_CNT	0 18	0 00	0 68	0 24	0 11	0 00	0 47	0 17	0 22	0,00	0 66	0 26	0 18	0 00	0 68	0 22
Residual Plots																

B.5. Lot CA0602

	Total rdm1c 1800mV 100ns				Total rdm2c 1800mV 100ns				Total rdm3c 1800mV 100ns				Total rdm4c 1800mV 100ns			
R²	0 23				0 15				0 10				0 15			
Significance	0 00				0 00				0 00				0 00			
F	43 15				25 69				15 24				25 47			
Critical F	3 03				3 03				3 03				3 03			
SEy	19.80				17 86				14 09				6 71			
	Beta	Sign	Coef	Conf Int	Beta	Sign	Coef	Conf Int	Beta	Sign	Coef	Conf Int	Beta	Sign	Coef	Conf Int
Const		0 03	6 07	5 42		0 26	2 82	4 88		0 00	7 17	3 85		0 00	21 99	1 83
GO_NR_W_0.28_0.4	0 39	0 00	1 81	0 63	0 31	0 00	1 26	0 57	0 23	0 00	0 72	0 45	0 04	0 63	0 05	0 22
CV_A_LIL	0 13	0 07	0 97	1 04	0 10	0 16	0 67	0 94	0 10	0 18	0 50	0 74	0 36	0 00	0 90	0 35
Residual Plots																

200 00
150 00
100 00
50 00
0 00
-50 00
200 00
150 00
100 00
50 00
0 00
-50 00
150 00
100 00
50 00
0 00
-50 00
60 00
40 00
20 00
0 00
-20 00