Simulation Results of an Efficient Defect Analysis Procedure

Olaf Stern, Hans-Joachim Wunderlich

Institute of Computer Structures, University of Siegen
Hölderlinstr. 3, D-57068 Siegen, Germany

Abstract

For obtaining a zero defect level, a high fault coverage with respect to the stuck-at fault model is often not sufficient as there are many defects that show a more complex behavior. In this paper, a method is presented for computing the occurrence probabilities of certain defects and the realistic fault coverage for test sets. The method is highly efficient as a pre-processing step is used for partitioning the layout and extracting the defects ranked in the order of their occurrence probabilities.

The method was applied to a public domain library where defects causing a complex faulty behavior are possible. The occurrence probability of these faults was computed, and the defect coverage for different test sets was determined.

1 Introduction

Safety-critical applications require extremely high quality standards which have to be guaranteed by refined testing techniques. Hence, very low *defect levels* are specified (less than 100 ppm) where defect level is defined as the percentage of defective parts that are misleadingly considered as good by the production test. High defect levels may be caused by a fault model which does not reflect the actual defect mechanisms [2,5,7,8,14,15,19,27]. In this case, even a high fault coverage does not ensure a high *product quality*, because the most likely physical defects might escape. The terms *defect coverage* or *realistic fault coverage* have been introduced for describing the quality of a test set T [25,26]. A realistic fault f changes the logic or timing behaviour of the circuit and is related to a set of defects D_f such that the occurrence of a defect $d \in D_f$ introduces the fault f into the circuit. The probability P(f) of fault f is identical with the occurrence probability $P(D_f)$. Let F be the set of all realistic faults, and $F(T) \subset F$ are the faults detected by test set T. The defect coverage of T is roughly estimated by: $DC(T) := \left(\sum_{f \in F(T)} P(f) \right) / \left(\sum_{f \in F} P(f) \right)$.

1.1 Objectives of the work

In recent years a variety of approaches were published for determining the set of realistic faults based on the circuit layout. The main obstacle to introducing these procedures into practice is the huge number of possible faults and defects causing a very high computing complexity.

In this paper a procedure is presented that extracts the realistic faults in the order of their occurrence probability. It is shown that the faults $F' \subset F$ with a high occurrence probability form a rather small subset of F, and their detection leads to sufficient defect coverage. One reason for that is the small number of defect mechanisms which dominate a fabrication process [12,13,14]. Moreover, extracting only the faults $F' \subset F$ leads to drastic saving in computing time for defect analysis such that small and medium sized layouts can be completely analyzed.

The main new idea for increasing the efficiency is to partition the layout into so called *elementary objects*. Computing defect probabilities is first restricted to these objects, and the results are refined and combined in a later step. The presented procedure has been applied to a public domain library [18], and the defect coverage obtained by test sets for 100% stuck-at fault coverage is discussed. The probability of such defects is computed whose delectability depends on test speed, driving strength at the inputs or the order of the applied test patterns.

1.2 State of the art

One of the first reports on the extraction of realistic faults is based on a stochastic insertion of defects into a layout description and analyzing the induced misbehavior [21]. This approach is known as *inductive fault analysis* [6,7,28]. As a large number of the inserted defects do not lead to a faulty behavior the efficiency can be increased if the analysis is restricted to so called *critical areas* [28] which have been introduced for yield estimation [17,23].

Later work tried to enhance both the precision and the efficiency of the analysis. In [9,26] the computations of the critical area was substituted by approximations such that more complex circuits could be handled. In [13] not

0-7803-2102-2/94 $4.00

only bridgings but also other defects as open lines were considered.

As the exact computation of critical areas is very expensive, several authors proposed a pre-processing of the circuit layout [4,10]. Gyves and Di used the concept of *susceptible sites* depending on the defect type. These are sites where a certain type of defect can cause a malfunction, and which restrict the area to be analyzed.

1.3 Organisation of the paper

In the next section an overview of the defect analysis tool EDEN (**E**fficient **D**efect **E**xtraction) is given, and it is shown how to pre-process the layout description for further analysis. In section 3 the algorithm is presented for extracting the defects in the decreasing order of their occurrence probabilities. In the following section these physical defects are mapped to electrical faults to be simulated. Simulation results are presented in section 5, and in section 6 two defects of an example layout are investigated. They have a rather significant occurrence probability, but show a complex malfunction, and they are hard to detect.

2 EDEN (Efficient Defect Extraction)

2.1 Defects

Defects are modifications of the circuit layout which can be distinguished in the following way:

1) Missing material in one object or in multiple objects of a single layer.
2) Additional material resulting in a connection of some objects of a single layer.
3) Additional material resulting in a overlapping of objects of different layers.

These types include the most relevant defects, for instance bridgings, opens, pin holes, and also some parameter variations. Based on process data and known defect mechanisms, the geometric alterations caused by a certain type of defects have to be determined manually. Moreover, we identify defects of a certain type if the same objects are affected in the same way, and hence they cause the same faulty topological structure. For instance defect A and B of figure 1a belong to the same class as both of them cause the faulty topology of figure 1b. But in figure 2 defect A causes topology b) and defect B causes topology c).

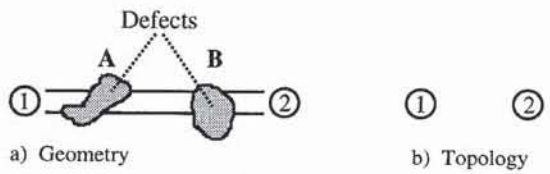


Figure 1: Two defects of the same class

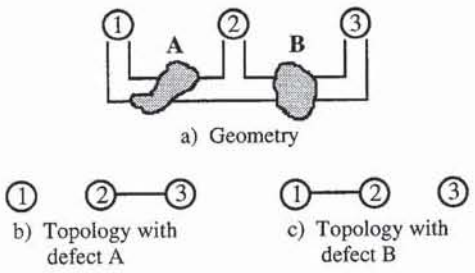


Figure 2: Two defects of different classes

Based on a layout description and data of statistic yield or defect monitoring, the program EDEN generates defects and their occurrence probabilities (figure 3). The main idea of this approach is layout pre-processing by a partitioning algorithm such that:

- a refined circuit extraction provides sufficient information for generating the faulty electrical functions,
- the fault probability can efficiently be calculated by using the concept of divide and conquer,
- all defect kinds are correctly mapped to the electrical netlist.

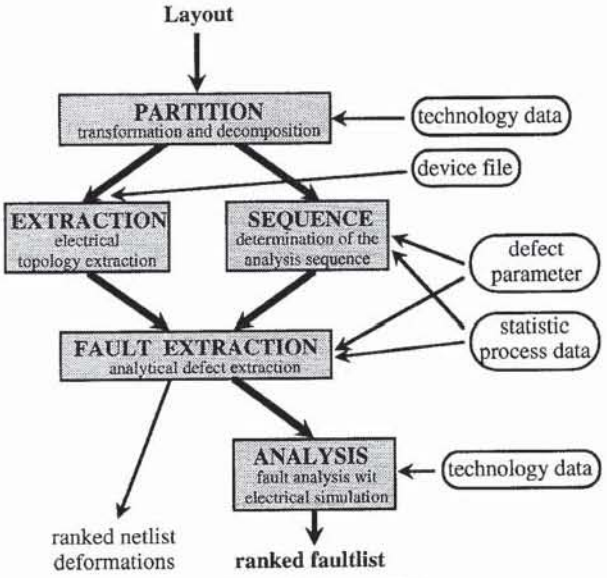


Figure 3: Basic flow of EDEN

2.2 Layout partitioning

The procedure *PARTITION* decomposes the layout in *elementary objects* which are defined by the following three conditions:

1) All defects of the same type and size with centres on the elementary object are equivalent.
2) The object is a rectangle.
3) The area of the object is maximum under condition 1 and 2.

Condition 2 and 3 allow an efficient calculation of the likelihood of faults by supporting a divide and conquer procedure. This calculation is based on the concept of *critical area* (see section 3.3) which is not tractable for arbitrary polygons but for rectangles. As an example, figure 4 shows the elementary objects for a single layout object. Defects on the elementary object 4 might cause a disconnection of all the other parts.

a) topology b) equivalent defect free electrical circuit

Figure 4: Example of elementary objects

Three kinds of elementary objects can be discriminated:

- *Redundancy rectangles* which have only a single or even no connection to a second object (e.g. object 3 in figure 4).
- *Connection rectangles* with exactly two connections (e.g. object 1).
- *Branch rectangles* with more than two connections (e.g. object 4).

The partition into elementary objects is not unique. The partitioning algorithm is actually implemented as follows. First a region is decomposed into disjoint rectangles of maximum size as shown in figure 5. Second, branches are identified. A rectangle with more than 2 direct neighbors (shadowed in figure 5) contains a branch and must recursively be decomposed into smaller rectangles while condition 3 of the definition is not violated (see figure 6). During those three steps also vertical connections resulting in contacts or transistors are considered.

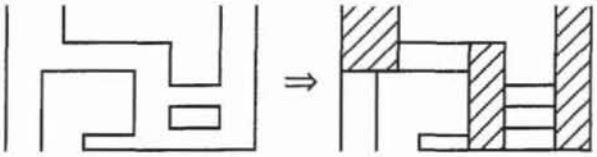


Figure 5: Partitioning of an object into maximum rectangles

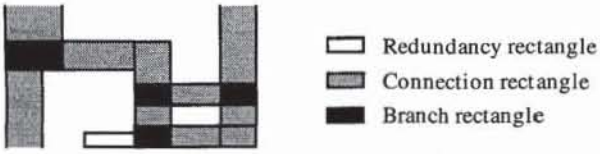
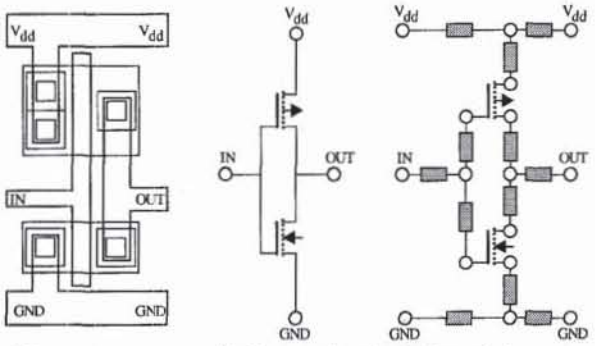


Figure 6: Region after complete partitioning

2.3 Extended netlist extraction

The computing efficiency is increased if the defect information at layout level is lifted to transistor level as early as possible. Therefore, the transistor netlist is refined such that every elementary object can exactly be assigned to one netlist element. In an extracted netlist branch rectangles are mapped to additional nodes whereas a path of connection rectangles is modeled by a simple edge usually representing parasitic resistors. In such an extended netlist all defects can exactly be located (see as an ex. fig. 2).

Figure 7 shows the simplified layout of a CMOS inverter, the result of a conventional netlist extractor, and the actually generated extended netlist by EDEN. For clearness capacitors are omitted.



a) Layout b) Conventional netlist c) Extended netlist

Figure 7: Layout and electrical netlists of a CMOS inverter

3 Ordering defect extraction

3.1 Analysis pairs

To handle more complex circuits it is necessary to concentrate on the defect mechanisms and defects with high occurrence probability. This restriction is justified since usually only a few defect kinds dominate in a fabrication process and many defects have a negligible occurrence probability [12,13,24]. Therefore, defects causing faults with high likelihood have to be analyzed first, which also reduces the test costs [22]. In this section a procedure for establishing an order and an estimation for the reached defect coverage is given.

For each layer i a list $RL^i := (R_{i,0}, .. , R_{i,n})$, $n \in IN$, of the elementary objects $R_{i,j}$, $j \in \{0, .. , n\}$ is generated such that $A(R_{i,j}) < A(R_{i,k})$ implies $k < j$ where $A(R)$ denotes the size of R. Based on process data, the probability $D_{i,j}$ of the defect type j on layer i per area unit is known. These data are used for instance for *yield modeling* and for the so called *product model* [23]. For the underlying technology process, a list $DT^i := (D_{i,0}, .. , D_{i,n})$ is generated which contains these probabilities in decreasing order (see figure 8).

A pair (R,D) of a rectangle $R \in RL^i$ and defect type $D \in DT^i$ describes a possible defect, and it is called *analysis pair*. For instance, if metal is the first layer, and the most

frequent defect type on this layer is missing material then $(R_{1,1}, D_{1,0})$ is an analysis pair describing the second rectangle of the RL-list for metal with this defect.

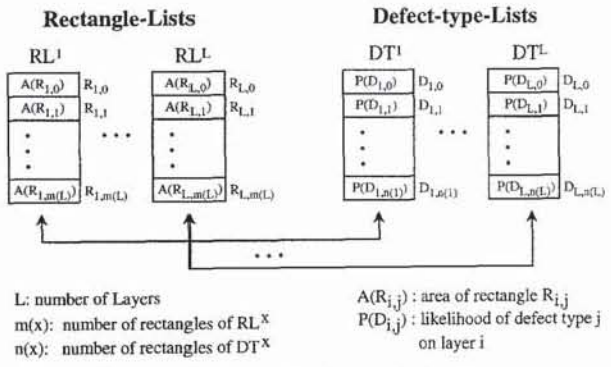


Figure 8: List of rectangles and defect types

An analysis pair provides a weight

$$W(R,D) := A(R) * P(D)$$

which is first rough estimation of the real defect probability. W(R,D) describes an upper bound, especially for defects affecting more than one rectangle, e.g. additional material resulting in a short. Here, just the weight for additional material connected with the rectangle R is computed.

A total weight is defined by

$$TW := \sum_{i=0}^{L} \left[\sum_{j=0}^{m(i)} A(R_{i,j}) \cdot \sum_{k=0}^{n(i)} P(D_{i,k}) \right]$$

where the notations of figure 8 are used. For a set AP of analysis pairs the weight is defined by:

$$W(AP) := \sum_{(R,D) \in AP} W(R,D).$$

An estimation of the defect coverage obtained by analyzing all pairs in AP is given by

$$DC := \frac{W(AP)}{TW}.$$

This is used as a stop criterion for generating analysis pairs. The procedure of figure 9 shows the computation of analysis pairs in the order of their weights W(R,D). If process data are not available all the $D_{i,j}$ have to be constant, and only the size of the objects is considered.

3.2 An example

The cells of the public domain library of the OCT-TOOLS [18] were completely analyzed. The most complex combinational cell is a 2:1-multiplexor realized as a CMOS complex gate. Figure 10 shows the layout. This circuit will serve as an example for the rest of this paper.

```
procedure SEQUE(RL¹,.., RLᴸ, DT¹,.., DTᴸ);
   AP_set := Ø; help_set := Ø;
   for all L do
      help_set += (RL,0 , DL,0);
   endfor;
   sort help_set;
   do
      AP_set += (first AP := (Ri,j , Di,j)
                  of help_set);
      help_set -= AP;
      if (Ri,j+1 , Di,j) exist and not in
                           help_set
         (add and sort) (Ri,j+1, D,j) into
                           help_set;
      endif;
      if (Ri,j , Di,j+1) exist and not in
                           help_set
         (add and sort) (Ri,j, Di,j+1) into
                           help_set;
      endif;
   until (estimated DC >= required defect
          coverage);
out AP_set;
endprocedure;
```

Figure 9: Determining analysis pairs

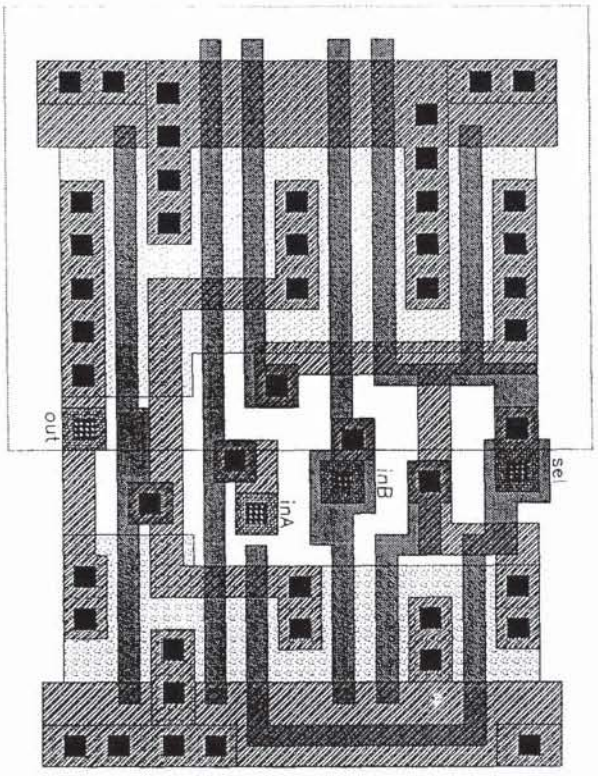


Figure 10: Example layout of a 2:1-multiplexor

As process data are confidential, for the analysis described here the data published in [24] were used. They are confirmed by the work of [12]. Figure 11 shows the esti-

mations of defect coverages DC under both the assumption that process data are not available, and based on process data.

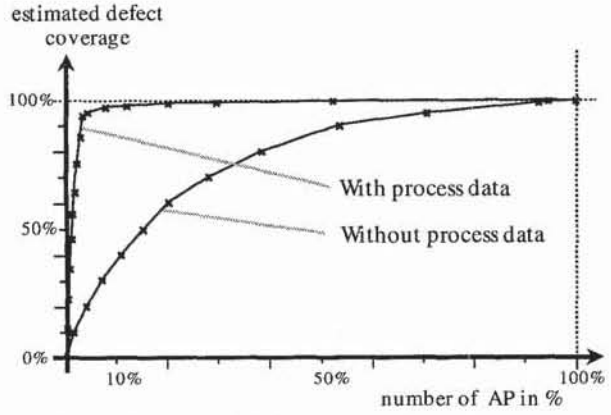


Figure 11: Estimated defect coverage by analysis pairs

Only a few percent of possible defects had to be extracted to reach over 90% weighted defect coverage. Using real process data reduces the number of analysis pairs to consider.

3.3 Critical area

Output of the procedure SEQUENCE (figure 3) is the sorted list of analysis pairs which have to be processed further to get exact defect probabilities. These probabilities are determined based on the concept of the *critical area* CA(d) for a defect with diameter d. It is the area, where placing the centre of such a defect will cause a fault.

Figure 12 shows the critical area for bridgings with respect to 2 elementary objects.

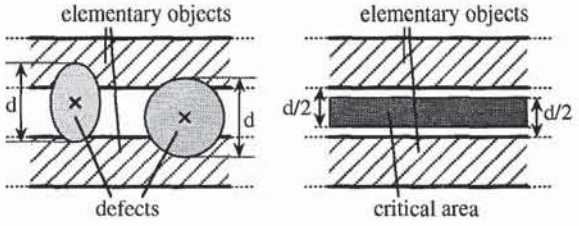


Figure 12: Two defects with diameter d, critical area and elementary objects

The average number λ of faults is estimated by

$$\lambda = \int_0^\infty CA(d) \cdot D(d)\, dd$$

$$= \overline{D} \cdot \int_0^\infty CA(d) \cdot h(d)\, dd.$$

where D(d) is the distribution of defect diameters, $\overline{D}$ is the average defect density, and

$$h(d) := \frac{D(d)}{\overline{D}}.$$

This leads to the *average critical area*

$$\overline{CA} := \int_0^\infty CA(d) \cdot h(d)\, dd.$$

This concept can be adapted to various defect distributions and yield models, for the analysis here the Gamma-distribution, and also the defect size distribution by Stapper are assumed [23].

The critical area depends on the shape and size of the involved objects. In the original concept CA were determined by approximations which are sufficient for yield modeling. But within cells these approximations may cause errors up to 100% [4]. Hence, a refined but efficient technique for computing critical area is required. In the presented approach an extended method of [4] is used. The refinements consist in a rather complex framework of formulas for computing CA at the boundaries. Here, geometric formulas that are numerically hard to evaluate are substituted by easily computable but still precise estimations. The used estimations allow closed formulas instead of an approximation by series, and take into account that usually several rectangles are connected. The ranked list of analysis pairs gives the objects and defect types for which critical areas are computed. As seen before, the critical area is proportional to the probability of such a defect.

4 Fault extraction

4.1 Faults at transistor level

The defect classes determined so far must be lifted to the electrical level. Equivalent defects that lead to the same faulty transistor netlist should be identified at an early stage of the process and their probabilities should be combined, whereas defects which do not have an electrical effect should be removed. For gaining efficiency this should be done before electrical simulation.

For each analysis pair (R,D) the corresponding defect is modeled in the netlist and the faulty behavior is determined. If two analysis pairs turn out to cause the same faulty transistor netlist their corresponding defect probabilities are combined. Furthermore, if two different topologies of a transistor netlist show the same behavior during electrical simulation their defect probabilities are united, too, resulting in a single value P(f) for the realistic fault f.

The input of this procedure is the ordered list of analysis pairs, output are realistic faults. The order of generating such a fault is determined by the weights of the analysis pairs, in the next section it is shown that these

weights and the realistic fault probabilities have a high correlation.

4.2 Example (continued)

The layout of the multiplexor circuit (figure 10) is partitioned into elementary objects which are mapped to the extended transistor netlist. The additional nodes correspond to branch rectangles (figure 13), bold nodes will be mentioned in the text.

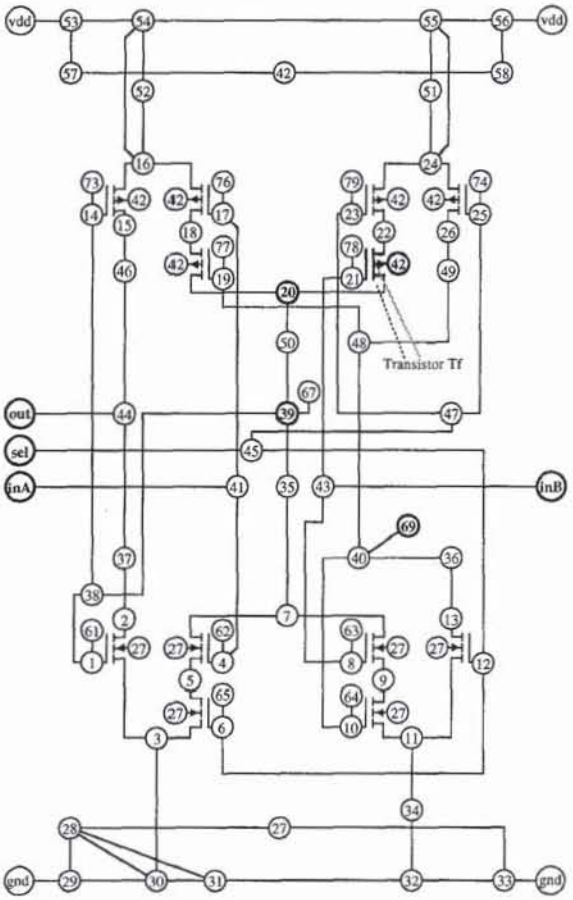


Figure 13: Extended netlist for the example circuit without resistors and capacitors.

In the approach presented so far, we first estimate defect coverage based on analysis pairs, afterwards we compute defect coverage based on critical areas, and the stop criterion is the estimated value before. Hence, its is specially important that estimated and exact values for the defect coverage are highly correlated. Figure 14 shows both curves fit very well for the example circuit, but it should be pointed out that this estimation cannot be used for determining the real weight of the faults, and only their ratios are established.

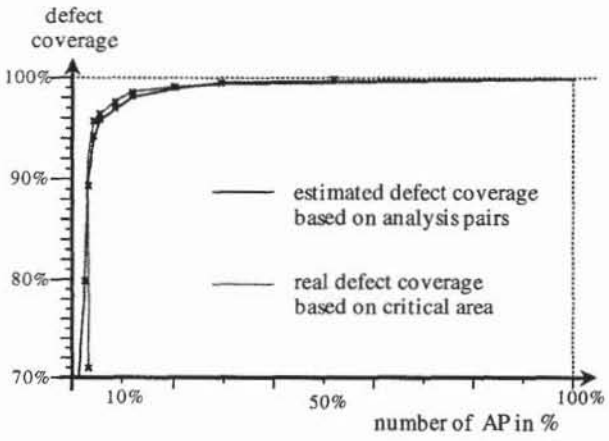


Figure 14: Estimated and real defect coverage.

5 Fault simulation

5.1 Waveform generation

The purpose of electrical fault simulation is both identifying equivalent defects and determining the defect coverage of a given test set. The first task needs a tremendous computing effort as the circuit behavior depends on many factors as the speed, the strength and the order of signals. Hence, this can only be done for small cells and a subset of the defects.

Moreover, the surroundings of a cell has to be considered, as it is impossible to stimulate the inputs directly. Usually the waveforms are generated by preceding cells, and responses are observed through following gates. The simulation model discussed in the sequel passes all input signals through a driver, and the output signal through an inverter (figure 15).

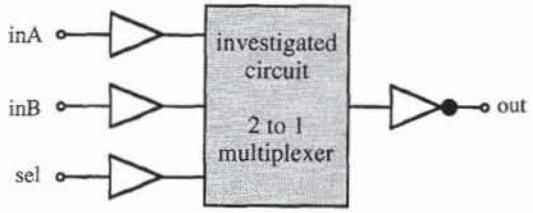


Figure 15: Simulation environment.

5.2 Defect coverage

If a test set is already generated, fault detection depends also on the ability of observing the behavior, which requires high capabilities of the tester equipment. Many faults may show a malfunction which is not deterministically detectable due to a limited observation interval of the tester, or a behavior which is still within the system specification.

Hence, we differentiate the effective defect coverage of a test set T in the following way:

$$- DC_{\ell}(T) := \frac{\sum_{f \in F_{\ell}(T)} P(f)}{\sum_{f \in F_{\ell}} P(f)}$$

is the coverage of defects which cause a logic fault.

- $DC_g(T)$ is the coverage of defects which cause a logic fault or a gross delay fault. F_g contains these faults, where gross delay faults have an additional delay of at least 25% of the maximum delay of the analysed circuit.
- DC(T) is the coverage of all defects F including small delay faults. There are also some short-timed spikes and glitches which may cause distortions during operating but are not guaranteed to be detected by the test equipment.

$DC_{\ell}(T)$ is obtainable by a standard tester, and the faults of $DC_g(T)$ are possibly detectable by a high performance equipment. The probability of detecting faults of DC(T) that produce only a small delay fault is very low.

$$DC^*(T) := \frac{\sum_{f \in F_{\ell}(T)} P(f)}{\sum_{f \in F} P(f)}$$

evaluats the portion of realistic faults surely detected by a standard test equipment.

5.3 Example (continued)

Based on the example layout, 214 modifications of the transistor netlist were extracted. These electrical faults form only about 50% of all possible defects, but detecting them would lead to a defect coverage of 99.9%.

A complete test set including all pattern sequences up to length 3 was applied using different driver strengths available in the used library. Table 1 shows the sum of the fault probabilities. The number in brackets denote logic faults which still show a dynamic behavior. First they produce a correct output value which is altered after a certain time period. These faults are only detectable by a low speed test.

6 Quality of test sets

In this section some defects are discussed which cause a hardly detectable fault behavior. During the analysis, well-known effects as a sequential behavior of stuck-open faults or some delays were observed. More interesting malfunctions include the dependency of fault detection on the driver strength at the input, and a dynamic malfunction where a correct output signal does not stay stable.

For the example circuit an exhaustive test set (ex), four deterministic test sets (d1 - d4), and three sets of random patterns were generated. The complex multiplexor cell was described by a netlist of single gates which were input to the deterministic test pattern generator SOCRATES [20]. The fault simulator FSILOS [3] was used for determining the stuck-at fault coverage of all the test sets. Size and stuck-at fault coverage are listed in table 2.

test set	# pattern	# covered stuck-at faults	stuck-at fault coverage
ex	8	22	100%
d1	4	22	100%
d2	4	22	100%
d3	4	22	100%
d4	4	22	100%
r1	4	13	59%
r2	4	18	82%
r3	8	18	82%

Table 2: Applied test sets.

6.1 Defect coverage and driving strength

First some statistic data are reported, then two defects are analyzed in deeper detail. All test sets were applied with three different driver strengths. The defect coverage DC_{ℓ} and DC^* are as defined before. Additionally, tables 3, 4 and 5 contain values for $\overline{DC_{\ell}}$ and $\overline{DC^*_{\ell}}$ where the faults are not weighted and are assumed to have all the same probability. This is the usual assumption, and it is only included for comparison reasons.

The usually estimated defect coverage $\overline{DC}$ and the realistic defect coverage DC differ significantly. Especially

driver strength	logic faults		logic + gross delay faults		all faults	
	number	$\sum_{f \in F_{\ell}} P(f)$	number	$\sum_{f \in F_g} P(f)$	number	$\sum_{f \in F} P(f)$
very high	127 (0)	1.9178e-06	139	1.9262e-06	151	5.2420e-06
high	130 (1)	1.9189e-06	139	1.9262e-06	151	5.2420e-06
medium	140 (2)	1.9262e-06	143	2.4337e-06	151	5.2420e-06
medium/low	143 (5)	2.1824e-06	151	5.2420e-06	152	5.2421e-06
low	150 (20)	4.9890e-06	151	5.2420e-06	151	5.2420e-06
very low	151 (10)	5.2420e-06	151	5.2420e-06	151	5.2420e-06

Table 1: Number and likelihood of realistic faults.

for medium and high driving strengths the conventional approaches overestimate the realistic value by more than 100%. None of the test sets covers all the realistic faults for all the driver strengths even if only logic faults are considered, despite the fact that five of the test sets obtain complete stuck-fault coverage. If a medium strength driver is applied, the effective fault coverage is in the average 40%. These results are worse with respect to the stuck-at fault model than some reported data [7] where fault detection of stuck-at test sets was determined by switch-level simulation. Test sets d1 , d2 and d3, d4, resp., only differ in the order of the applied patterns, but show significant differences in the defect coverage.

test set	#F$_\ell$	DC$_\ell$	$\overline{\text{DC}_\ell}$	**DC***	$\overline{\text{DC}^*}$
ex	127	99.94	97.69	**36.58**	84.10
d1	126	99.94	96.92	**36.58**	83.44
d2	129	99.95	99.23	**36.59**	85.43
d3	117	86.09	90.00	**31.51**	77.48
d4	119	86.09	91.53	**31.51**	78.80
r1	80	51.66	61.53	**18.91**	52.98
r2	68	49.94	52.30	**18.28**	45.03
r3	122	99.69	93.84	**36.49**	80.79

Table 3: Defect coverage with high driver strength.

test set	#F$_\ell$	DC$_\ell$	$\overline{\text{DC}_\ell}$	**DC***	$\overline{\text{DC}^*}$
ex	135	99.66	96.42	**36.62**	89.40
d1	135	99.91	96.42	**36.71**	89.40
d2	137	99.82	97.85	**36.68**	90.72
d3	125	86.02	89.28	**31.61**	82.78
d4	126	85.85	90.00	**31.55**	83.44
r1	77	51.36	55.00	**18.87**	50.99
r2	69	49.79	49.28	**18.29**	45.69
r3	125	99.37	89.28	**36.52**	82.78

Table 4: Defect coverage with medium driver strength.

test set	#F$_\ell$	DC$_\ell$	$\overline{\text{DC}_\ell}$	**DC***	$\overline{\text{DC}^*}$
ex	151	100.00	100.00	**100.00**	100.00
d1	148	99.99	98.01	**99.99**	98.01
d2	151	100.00	100.00	**100.00**	100.00
d3	144	95.12	95.36	**95.12**	95.36
d4	147	95.12	97.35	**95.12**	97.35
r1	96	77.53	63.57	**77.53**	63.57
r2	104	52.43	68.87	**52.43**	68.87
r3	144	99.82	95.36	**99.82**	95.36

Table 5: Defect coverage with low driver strength.

The result concerning the random test set r3 also indicates that stuck-at fault coverage does not reflect the coverage of realistic faults sufficiently. Here, only a stuck-at fault coverage of 82% was obtained, but defect coverage was higher than obtainable by the deterministic test sets d3 and d4 with 100% stuck-at coverage. A similar result was observed for the other investigated cells of the library and by measuring actually produced dies [16].

As seen in table 1, the number of faults that cause only a delay is decreasing if the driver strength is reduced, and the number of logic faults is growing. Hence, testing is becoming easier by lowering the strength of the input drivers as also observed in [11]. In numbers, only 130 faults out of 151 faults cause a logic malfunction if the driver strength is high, but using low strength drivers all 151 faults are detectable statically. As the driving strength is pattern dependent, some faults may be overlooked during testing, moreover, they may reduce reliability (see also [11]). For all the test sets the realistic fault coverage DC is increasing by reducing driving strength, and two of them (d2 and ex) lead to complete realistic fault coverage. In the presented example, the variations of the driver strength are uniform for all inputs, the results looks similarly if different inputs are driven with different strengths.

6.2 An example defect

The test set d2 (table 6) was applied to a gate oxide defect within the p-channel transistor such that the gate is connected with the N-tub, which is node 42 in the extended netlist. This realistic fault is number 2 of the ranking list, and its probability is P(f) = 6.21e-07.

inA	inB	sel	out
0	1	0	0
0	1	1	1
1	0	1	0
1	0	0	1

Table 6: Test set d2.

Figure 16 is the output plot of the electrical simulation with SPICE for the defect free circuit. Besides the voltage of node out and node inB also the graph for node 20 is shown, which is the drain of the defective transistor.

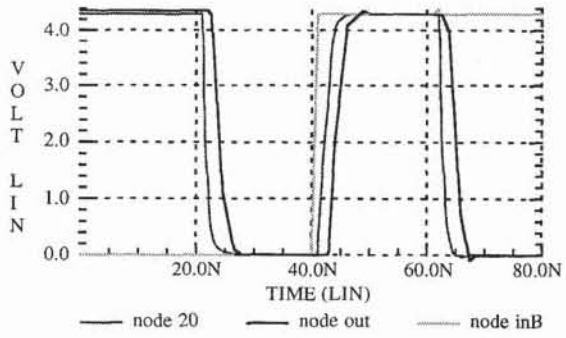
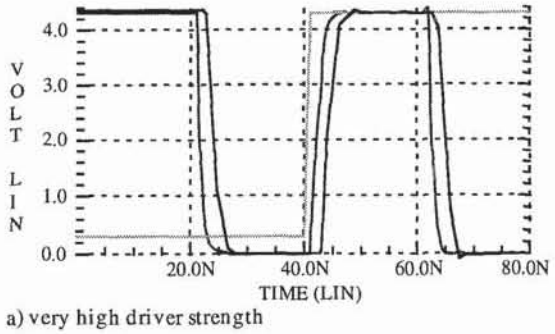
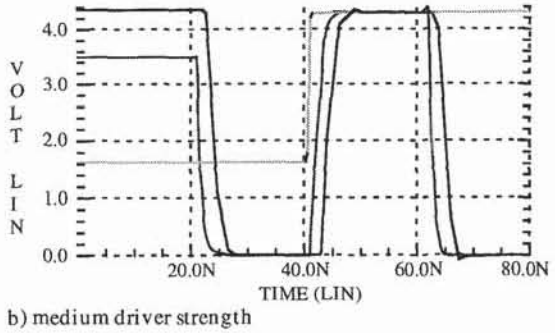


Figure 16: Electrical simulation of the defect free-circuit.

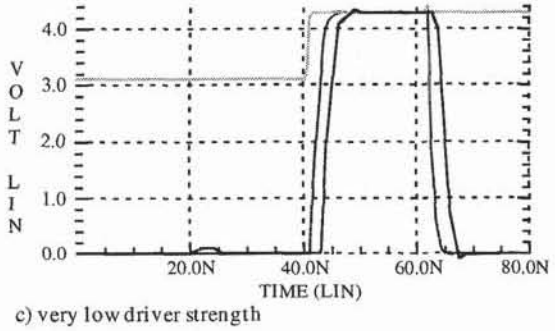
Figures 17a - 17c show the simulation results for the defect circuit with falling driving strength. The bridge divides the voltage such that for strong input driving the resulting value of node 20 is in the range of 0.4 V and 1.6 V, and the transistor T_f does not switch, but reducing the driver strength holds node 20 on a low potential, and the output *out* is definitely on GND. For stronger drivers at the input, the fault does not cause a notable delay ($0 \leq$ 0.1ns), and seems only to be detectable by iddq-testing. This supports the results reported in [15] where a combination of logic testing, delay testing and iddq-testing is proposed for obtaining maximum product quality.



a) very high driver strength



b) medium driver strength



c) very low driver strength

Figure 17: Electrical simulation of the defect circuit with decreasing driver strength for test set d2.

6.3 An example of a dynamic fault

A bridging between node inB and node 69 may be caused by a defect of the metal layer, and has probability p(f) = 6.12e-9 which is rank 17 in the fault list. If test set d3 (see table 7) is applied, the shorted nodes inB and 69 get complementary values at the third pattern, resulting in a voltage of 2.3 V at node 69.

inA	inB	sel	out
1	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0

Table 7: Test set d3.

The small value of 2.3 V makes the following transistor conducting with a considerable resistance such that node 39 which is connected to the gates of the output inverter looses its capacity slowly. Figure 18 shows that the correct output is generated for approximately 20 ns before the signal falls to the erroneous low level. The immediate transition is due to the inverter at the output.

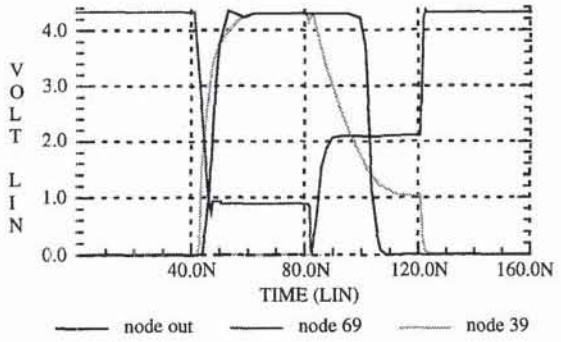


Figure 18: Simulation results for a bridging for node inB and node 69 using test set d3 and low driver strength.

This defect did not cause any other faulty behavior, and cannot be detected by usual high speed test. Moreover, even if testing is done at low speed this fault may escape and cause a wrong output later as during operation the same pattern may be repeated several times. Similar to the fault described in the previous section, this fault produces a considerable iddq-current and can be detected by iddq-testing on principle.

Conclusions

A method was presented for efficient determining the realistic fault coverage based on the circuit layout. The layout is partitioned into elementary objects which are specially suited for defect analysis. The faults are extracted and ranked in the order of their occurrence probability.

A CMOS library was investigated using this tool, and several results were obtained. The defect coverage of a test set generated for stuck-at faults of the gate level equivalent is not always superior to random test sets. For all test sets the defect coverage increases if the strength of the input driver is reduced. There are faults which only have an effect for a certain driving strength, other faults cause instable outputs, delays or sequential behavior.

Using the tool presented, the defects causing these faults are identified and their occurrence probabilities are computed. Further applications are in the field of fault diagnosis and layout synthesis for testability.

References

[1] E.M.J.G. Bruls: Reliability aspects of defect analysis, Proc. of European Test Conference, Rotterdam, NL, April 1994, pp. 17-26.

[2] Kenneth M. Butler, M. Ray Mercer: Quantifying Non-Target Defect Detection by Target Fault Test Sets, Proc. of European Test Conference, Munich, April 1991, pp. 91-100.

[3] Cadence Design System, Cadence SILOS II , Integrated IC Design System, Reference Manual, 1993.

[4] F. Corsi, S. Martino, C. Marzocca, R. Tangorra, C. Baroni, M.Buraschi: Critical Area for Finite Length Conductors, Microelectronics & Reliability, Vol. 32, No. 11, 1992, pp. 1539-1544.

[5] F. Corsi, C. Morandi: Inductive fault analysis revisited, IEE Proceedings-G, Vol. 138, No. 2, April 1991, pp. 253-263.

[6] R.J. Evans, W.R. Moore: Layout Based Testing of CMOS Logic, Department of Engineering Science, University of Oxford, December 1989.

[7] F. Joel Ferguson, John P. Shen: A CMOS Fault Extractor for Inductive Fault Analysis, IEEE Transactions on CAD, Vol. 7, No. 11, November 1988, pp. 1181-1194.

[8] J. Galiay, Y. Crouzet, M. Vergniault: Physical Versus Logical Models MOS LSI Circuits: Impact On Their Testability, IEEE Transactions on Computers, Vol. C-29, No. 6, June 1980, pp. 527-531.

[9] J.A. Grácio, P.A. Bicudo, N.N. Rua, A.M. Oliveira, C.F.B. Almeida, J.P. Teixeira: Test Preparation and Fault Analysis using a Bottom-Up Methodology, Proc. of European Test Conference, Paris, April 1989, pp. 168-174.

[10] Jose Pineda de Gyvez, Chennian Di: IC Defect Sensitivity for Footprint-Type Spot Defects, IEEE Transactions on CAD, Vol. 11, No. 5, May 1992, pp. 638-658.

[11] Hong Hao, Edward J. McCluskey: Very-Low-Voltage Testing for Weak CMOS Logic ICs, Proc. of IEEE International Test Conference, Baltimore, October 1993, pp. 275-284.

[12] Christopher Hess, Larg H. Weiland: Teststrukturen zur Bestimmung von Defektparametern für hochintegrierte Schaltungen, Diplomarbeit, Universität Karlsruhe, Institut für Rechnerentwurf und Fehlertoleranz, 1992.

[13] Marcel Jacomet: FANTASTIC: Toward a Powerful Fault Analysis and Test Pattern Generator for Integrated Circuits, Proc. of IEEE International Test Conference, Washington DC, August 1989, pp. 633-642.

[14] Wojciech Maly: Realistic Fault Modeling for VLSI Testing, Proc. of IEEE DAC, Miami, June 1987, pp. 173-180.

[15] Peter C. Maxwell, Robert C. Aitken, Vic Johansen, Inshen Chiang: The Effect of Different Test Sets on Quality Level Prediction: When is 80% Better than 90%?, Proc. of IEEE International Test Conference, Nashville TN, October 1991.

[16] Peter C. Maxwell, Hans-Joachim Wunderlich: The Effectiveness of Different Test Sets for PLAs, Proc. of European Design Automation Conference, Glasgow March 1990.

[17] Phil Nigh, Wojciech Maly: Layout-Driven Test Generation, ICCAD 89, Santa Clara California, 1989.

[18] OCTTOOLS-5.2 User's Guide, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720, May 1993.

[19] M. Saraiva, P. Casimiro, M. Santos, J.T. Sousa, F. Gonçalves, I. Teixeira, J.P. Teixeira: Physical DFT for High Coverage of Realistic Faults, Proc. of IEEE International Test Conference, Baltimore, September 1992, pp. 642-651.

[20] Michael H. Schulz, Erwin Trischler, Thomas M. Sarfert: SOCRATES: A Highly Efficient Automatic Test Pattern Generation System, IEEE Transactions on CAD, Vol. 7, No. 1, 1988, pp.126-137.

[21] J.P. Shen, W. Maly, F.J. Ferguson: Inductive Fault Analysis of nMOS and CMOS Integrated Circuits, Research Report No. CMUCAD-85-51, Carnegie-Mellon-University, August 1985.

[22] Gerald Spiegel: Optimized Test Cost using Fault Probabilities, IEEE, Proc. of European Test Conference, Rotterdam, April 1993, pp. 188-193.

[23] C.H. Stapper: Modeling of integrated circuits defect sensitives, IBM Journal of Research and Development, Vol. 27, November 1983, pp 549-557.

[24] C.H. Stapper: Improved Yield Models for Fault-Tolerant Random-Access Menory Chips, IEEE, Proc. Workshop on Defect and Fault Tolerance VLSI Systems, Hidden Valley, Pennsylvania, November 1989, pp. 46-59.

[25] O. Stern, H.-J. Wunderlich: Erfassung und Modellierung komplexer Funktionsfehler in Mikroelek-tronik-Bauelementen, ITG-Fachbericht 119, VDE-Verlag, ITG-Fachtagung: Mikroelektronik für die Informationstechnik, Stuttgart, March 1992, pp. 117-122.

[26] J.P. Teixeira, I.C. Teixeira, C.F.B. Almeida, F.M. Conçalves, J. Conçalves, R. Crespo: A Strategy for Testability Enhancement at Layout Level, Proc. of European Design Automation Conference, Glasgow, March 1991, pp. 413-417.

[27] R.L. Wadsack: Fault Modeling and Logic Simulation of CMOS and MOS Integrated Circuits, Bell System Technical Journal, No. 4, May 1978.

[28] Duncan Moore Henry Walker: Yield Simulation for Integrated Circuits, Carnegie-Mellon-University, Kluwer Academic Publishers, 1987.