VLSI DESIGN
1998, Vol. 7, No. 2, pp. 163–176
Reprints available directly from the publisher
Photocopying permitted by license only

Realistic Fault Modeling and Extraction of Multiple Bridging and Break Faults

GERALD SPIEGEL and ALBRECHT P. STROELE*

Institute of Computer Design and Fault Tolerance (Prof. Dr.-Ing. D. Schmid), University of Karlsruhe, Germany

(Received 1 August 1996)

Fault sets that accurately describe physical failures are required for efficient pattern generation and fault coverage evaluation. The fault model presented in this paper uniquely describes all structural changes in the transistor net list that can be caused by spot defects, including bridging faults that connect more than two nets, break faults that break a net into more than two parts, and compound faults. The developed analysis method extracts the comprehensive set of realistic faults from the layout of CMOS ICs and for each fault computes the probability of occurrence. The results obtained by the tool REFLEX show that bridging faults connecting more than two nets account for a significant portion of all faults and cannot be neglected.

Keywords: Bridging faults, break faults, fault modeling, fault extraction, multiple faults, spot defects

1. INTRODUCTION

Today's semiconductor manufacturing technologies cannot guarantee that all produced chips completely satisfy the specification. Therefore the chips have to be tested before they are delivered to the customer. Usually the number of possible physical failures is very large, and it is impractical to consider all of them explicitly. Fault models describe the failures at a more abstract level and thus provide the base for efficient test pattern generation and fault coverage evaluation. But two problems are associated with fault models. On the one hand, a fault model should cover all the physical failures that actually occur. If a physical failure is not represented by a fault, no test is generated for it, and faulty devices may be classified fault-free which leads to a loss of product quality. On the other hand, an abstract fault model may include some "artificial" faults that do not correspond to any physical failures in the implemented circuit. Tests for "artificial" faults increase the test application time without improving product quality.

The most commonly used fault model is the stuck-at fault model [1]. The growing density of integration in CMOS technology, however, increases the importance of bridging and break

*Corresponding author.

faults [2], [3], and many of these are not covered by the stuck-at fault model. If we look only at the transistor net list or at the gate level description of a circuit, a large variety of bridging and break faults seems possible. In order to determine which of these faults can really occur, the circuit must be analyzed at layout level.

Inductive Fault Analysis [4] is a systematic approach to generate an accurate fault set. The circuit layout is analyzed to obtain all possible deviations from the intended structure. Hence, the extracted fault set merely consists of faults which actually occur due to fabrication defects. Combining geometrical information and statistical data on defects, the probability of occurrence is determined for each fault. These fault probabilities can be used for yield estimation (e.g., [5], [6], [7]), physical design for testability [8], and test optimization [9].

Several methods of inductive fault analysis have been developed in the past few years. The FXT tool [10] is based on statistical simulation which is, however, very time consuming and less accurate compared to analytical methods. FANTESTIC [11] is an analytical approach which takes up the critical area concept. The critical area is the layout area in which a defect must fall to cause a fault [12]. Since critical area calculation is not a trivial task, FANTESTIC is limited to simple layout structures and uses coarse approximations in favor of efficient geometric operations [13] also describes procedures to compute critical areas and to derive test patterns. The CARAFE system [14] is able to extract possible bridging and break faults from arbitrary Manhattan layouts. However, the analysis is restricted to bridges between two nodes and breaks which cut a node in exactly two subnodes. LIFT [8] an ACRIT [15] use similar simplifications in critical area calculation and fault modeling. More recently, a net-oriented approach was proposed [16] where for each net separately the possible bridging faults to neighboring nets are extracted. But as each bridging fault involves at least two nets, faults are extracted multiple times and must be collapsed afterwards when the fault lists are merged. This approach was also applied to break faults [17]. To handle more complex circuits, a ranking of defects was proposed [18]. Defects that cause faults more likely are analyzed first.

In summary, the fault models used by previous fault extraction methods have three major limitations:

- Bridging faults have been restricted to the connection of only two nets.
- A net which is affected by a break fault has been assumed to be cut into exactly two subnets. Some approaches have not differentiated between different positions of a break fault in the net.
- Correlations between faults have not been considered. (Two faults are *correlated* if there exists a single defect which can cause both faults simultaneously).

Examples show that several multiple-net bridging faults are not detected by tests for 2-net bridging faults (see Section 5). So the multiple-net bridging faults which occur with nonnegligible probability should be considered explicitly during test pattern generation and fault coverage evaluation. The location and the multiplicity of break faults also has an impact on the faulty behavior. Even break faults that affect the same net may require different pairs of test patterns.

A situation with two correlated bridging faults is shown in Figure 1. A single "missing insulator" defect causes two bridges between different nets. Hence the occurrence of these faults is not

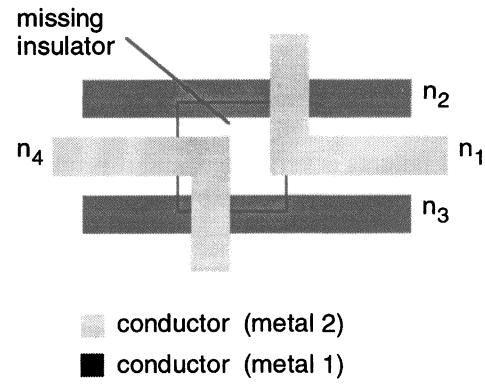


FIGURE 1 Correlation between faults.

statistically independent, and the probability that at least one of them occurs (making the circuit faulty) cannot be calculated accurately without knowing the correlation between these faults.

Similarly, correlated break faults exist if more than one net is affected by a single defect. In general, correlated faults make it difficult to calculate fault probabilities at the electrical level.

In this paper, a realistic fault model is presented which overcomes the above-mentioned problems. The proposed fault model facilitates the unique representation of all possible changes in the structure of a circuit at the electrical level. The faults are modelled such that each defect in the layout can cause at most one fault, and the faults are not correlated. We also present a unified approach to extract the complete set of single and multiple bridging and break faults from the layout of a circuit. The probability of occurrence is determined for each fault using a novel method for fast critical area calculation. The presented approach is applicable to arbitrary layouts with Manhattan geometry.

The paper is organized as follows. After introducing the representation of a circuit and its faults on layout and on transistor level, the improved realistic fault model is described and the fault extraction method is presented. Then, experimental results are shown, and a partitioning approach for the efficient processing of large circuit layouts is proposed.

2. CIRCUIT AND FAULT REPRESENTATION AT DIFFERENT LEVELS OF ABSTRACTION

As a bottom-up process, fault extraction starts at the layout level. We partition the geometric layout structures into rectangular regions according to the Corner-Stitching structure [19]. Each layout object is characterized by the coordinates of its corners, a layer attribute (e.g., polysilicon, insulator, metal, diffusion) and pointers to adjacent objects in a horizontal (intra-layer) or vertical (inter-layer) direction. Using a circuit extraction procedure, a label is assigned to all conducting and semiconducting objects that indicates which net they belong to.

The electrically conductive connections between layout objects are modelled by a *connectivity graph* $G=(V, E)$. Its vertices V represent the layout objects of conducting or semiconducting material. The edge set E contains an edge $\{v_1, v_2\}$ if and only if the layout objects represented by v_1 and v_2 are connected directly, i.e., not only via other objects.

On the electrical level, a circuit is described as a set of electrical nodes $C = \{c_1, \ldots, c_q\}$, a set of transistors $T = \{t_1, \ldots, t_r\}$, and a set of nets $N = \{n_1, \ldots, n_s\}$. A node c_i is a transistor terminal, a primary input or output, or a power supply junction. The terminals of a transistor t are denoted by $s(t)$ for source, $d(t)$ for drain, and $g(t)$ for gate. (Generally, source and drain of a transistor are given by the current direction. For the purpose of fault extraction, however, the distinction of source and drain is not relevant). At the electrical level, each net is defined by a subset of the nodes of C, namely the electrically connected nodes. The nets of a circuit are pairwise disjoint, they form a partition of C. At the layout level, each net corresponds to a connected component (maximal connected subgraph) of the connectivity graph G.

Due to deviations and irregularities in the manufacturing process, local perturbations, called defects, occur. In this paper, we consider *spot defects* with a size which is comparable to the minimum feature size of the layout. A multitude of different defect mechanisms describes how defects are produced. The most important types of defects are "extra material" and "missing material" in a single conducting or isolating layer (this also includes defects which have an impact on the connectivity of vias between different conducting layers). These defects can be modelled as additional or missing layout objects. Defects are statistically distributed over the whole chip. Their frequency and the distribution of defect sizes strongly depend on the type of the defect.

Defects may cause local deviations in the connectivity of conducting and semiconducting layout objects. These deviations are called *connectivity faults*. In section 4, connectivity faults will be defined such that each defect causes at most one connectivity fault. But generally, many different defects can lead to the same connectivity fault.

Finally, a connectivity fault can cause a change in the circuit structure at the electrical level and in this way manifest itself as a faulty behavior. Different connectivity faults can lead to the same fault in the transistor net list. But of course, there are also connectivity faults that do not influence the net list, for example a fault that connects two parts of the same net or a fault that interrupts a ring shaped net at only one position. Figure 2 summarizes these relationships.

In the following, a square defect shape is assumed rather than a circular or octagonal one because the geometric algorithms are far easier (see also [15]). This approximation results in fault probabilities that are slightly too large, but the completeness of the extracted fault set is not affected.

3. FAULT MODEL

The fault extraction approach presented in this paper aims at faults at the electrical level. The fault model used here consists of three fault types: Bridging faults, break faults, and compound faults. Transistor stuck-on and stuck-open faults are also modelled as bridging and break faults, respectively. All these faults have an influence on the partition of the set of electrical nodes into separated nets. Bridging faults merge some nets, break faults split some nets, and compound faults do both.

DEFINITION 1 Let N be the set of nets of a circuit, and let $BF = \{N_1,\ldots,N_v\}$ be a set of pairwise disjoint subsets of nets, $N_j \subset N, j = 1, 2, \ldots, v$, with a least two nets in each subset N_j. BF is a *bridging fault* if a single spot defect can cause all nets of each subset $N_j \in BF$ to be electrically connected.

Example The CMOS AND gate in Figure 3 has the nets $n_1 = \{i_1, g(t_1), g(t_4)\}$, $n_2 = \{i_2, g(t_2), g(t_5)\}$, $n_3 = \{d(t_6), d(t_5), \text{GND}\}$, $n_4 = \{s(t_1), s(t_2), s(t_3), V_{DD}\}$, $n_5 = \{d(t_1), d(t_2), s(t_4), g(t_3), g(t_6)\}$, $n_6 = \{d(t_4), s(t_5)\}$ and $n_7 = \{d(t_3), s(t_6), \text{out}\}$.

In a single defect causes the connection of nets n_1, n_2, and n_3 and the connection of n_4 and n_5, the resulting bridging fault is $BF = \{\{n_1, n_2, n_3\}, \{n_4, n_5\}\}$.

DEFINITION 2 Let $UF = \{\prod_1(n_1), \ldots, \prod_w(n_w)\}$ be a set of partitions for the nets $n_i \in N$ with $|\prod_i(n_i)| \geq 2$ for $i = 1, 2, \ldots, w$. UF is a *break fault* if a single spot defect can cause each net $n_i \in \{n_1, \ldots, n_w\}$ to be divided into two or more disconnected subnets according to $\prod_i(n_i)$. All other nets are preserved.

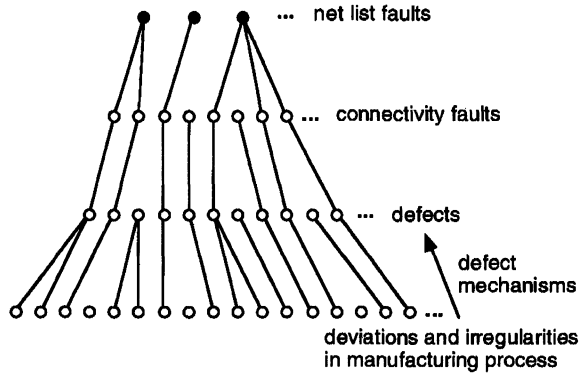


FIGURE 2 Defects, connectivity faults, and net list faults.

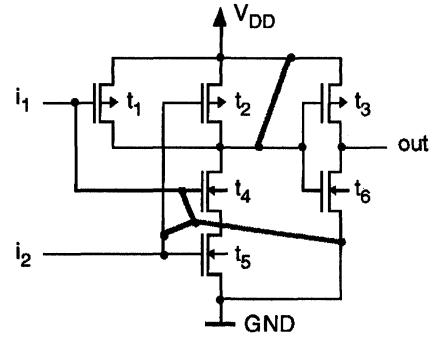


FIGURE 3 Bridging fault in a CMOS AND gate.

Example Figure 4 shows the CMOS AND gate again. Now a single defect causes a break which cuts nets n_2 and n_5 into two and three subnets, respectively. The resulting break fault is $UF = \{\prod_1(n_2), \prod_2(n_5)\}$ with $\prod_1(n_2) = \{\{i_2, g(t_5)\}, \{g(t_2)\}\}$ and $\prod_2(n_5) = \{\{d(t_1), s(t_4)\}, \{d(t_2), \{g(t_3), g(t_6)\}\}\}$.

A "missing polysilicon" defect in the region of a transistor channel can cause a short between source and drain as well as a break of a signal line. As a consequence, a bridging and a break fault may occur simultaneously due to a single defect. To avoid correlation between these faults, this case has to be modelled separately.

DEFINITION 3 A combination of a bridging fault and a break fault, $CF = \{BF, UF\}$, is a *compound fault* if a single spot defect can cause the bridging fault BF and the break fault UF together.

Example Figure 5 shows a CMOS inverter and its layout. A "missing polysilicon" defect in the region of the n-transistor channel leads to a stuck-on n-transistor and a floating gate of the p-transistor.

The fault model of definitions 1 to 3 provides a unique way to describe all structural deviations that may occur in the circuit structure at the electrical level. Moreover, all the faults are uncorrelated, they occur statistically independent if the underlying defects occur statistically independent.

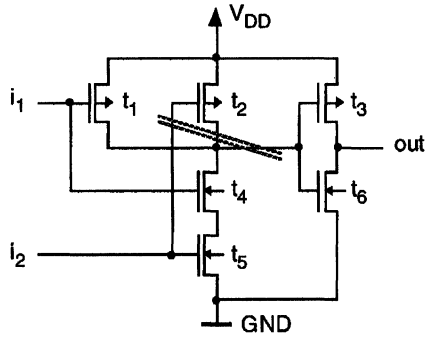


FIGURE 4 Break fault in a CMOS AND gate.

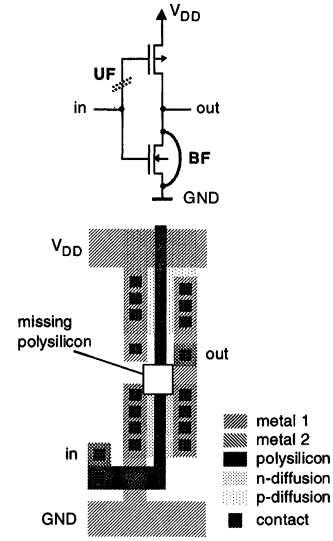


FIGURE 5 Compound fault in a CMOS inverter.

4. EXTRACTION OF REALISTIC FAULTS

In the context of testing, defects that do not lead to a connectivity fault and also connectivity faults that do not lead to a net list fault are not relevant since they do not affect the behavior of the circuit (possibly apart from reliability). So we can focus on the faults in the net list and in particular on those faults that occur with nonzero probabilities (*realistic faults*). The advantage of this more abstract view of faults is that the number of faults that must be considered is by far smaller than the number of the connectivity faults and the number of defects.

4.1. Overview

In order to extract the realistic net list faults of a circuit, the following problem must be solved:

Given:
- Description of the layout
- Set of defect mechanisms
- Defect statistics

Find:
- Complete set of realistic faults (according to the fault model of section 3) and

- probability of occurrence, $P(F)$, of each extracted fault F

Figure 6 gives an overview of the developed method. It begins with extracting the net list of the fault-free circuit from the layout data. The set of rectangular layout objects is determined, and each conducting object is marked with the number of the net it belongs to.

The following two steps are repeated for all the defect mechanisms that lead to extra material or missing material in one of the layers and also for different defect sizes. Using the knowledge about possible defects, the layout is analyzed regarding connectivity faults. First, only *fault primitives* are considered. These are

- undesigned connections of two objects in the same layer
- undesigned connections of two objects in different layers (through missing insulator)
- disruptions of one object
- disconnections of two objects in the same layer
- disconnections of two objects in different layers

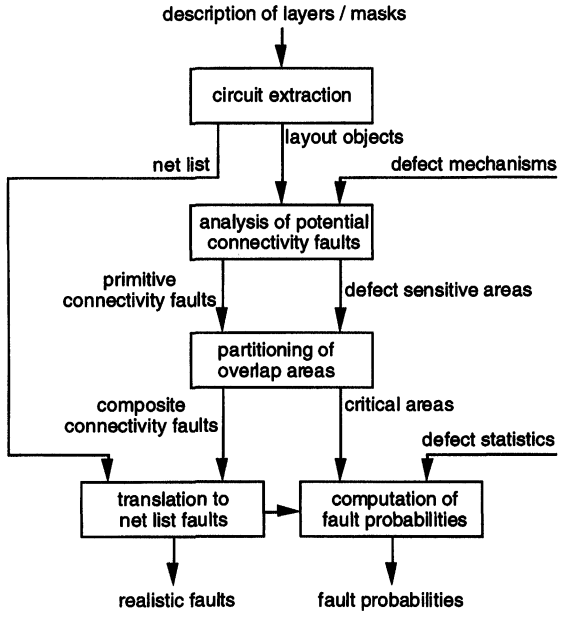


FIGURE 6 Fault extraction.

and some special cases that depend on the applied technology, e.g., disruption of a transistor gate. For each of these fault primitives the *defect sensitive area* is determined, that is the area in which the center of a defect of a given size must fall to cause the considered fault primitive (possibly in combination with other fault primitives).

If the center of a defect falls in a region where k defect sensitive areas overlap, this defect causes all the corresponding deviations in connectivity simultaneously. The result is a *connectivity fault* f including k fault primitives, $f=\{f_1,\ldots,f_k\}$. This concept of a connectivity fault as a nonempty set of fault primitives ensures that the effect of each defect can be described by exactly one connectivity fault. The *critical area* of a connectivity fault f, $\mathrm{ca}(f, s)$, is the area in which a defect of size s must fall to cause this and only this connectivity fault. Figure 7 shows an example with two fault primitives f_1 and f_2 and the connectivity faults $\{f_1\}$, $\{f_2\}$, and $\{f_1, f_2\}$. The example considers defects of a fixed size.

In principle, the critical area $ca\ (f, s)$ has to be determined for all possible defect sizes s. To compute the probability of occurrence, $P(f)$, of a connectivity fault f, its critical areas are weighted

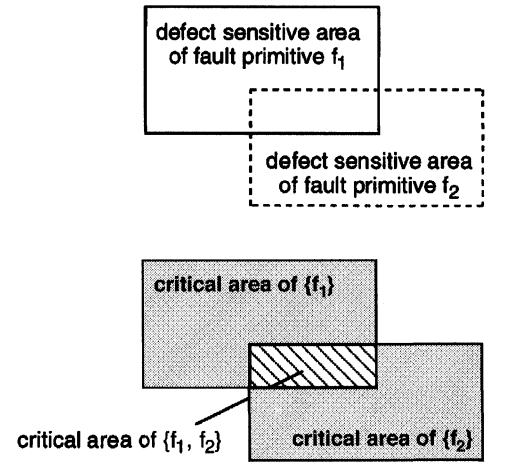


FIGURE 7 Defect sensitive area of fault primitives and critical area of connectivity faults.

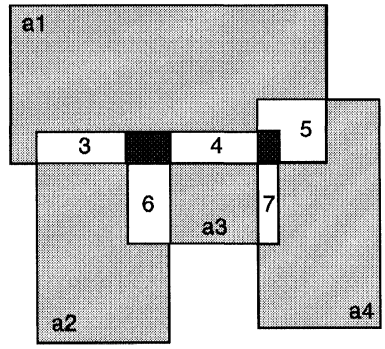
according to the distribution of the defect sizes, $H_{dm}(s)$, multiplied by the defect density D_{dm} of the considered defect mechanism dm, and the contributions of all defect mechanisms are added,

$$P(f) = \sum_{dm}(D_{dm} \cdot \int_0^\infty ca_{dm}(f,s) \cdot H_{dm}(s)ds) \quad (1)$$

The integral can be approximated by evaluating the critical area for some defect sizes exactly and then applying Simpson's Rule.

Finally, the connectivity faults are translated to net list faults, and the probabilities of these net list faults are computed. The translation can be done efficiently by a circuit extraction process that is restricted to the neighborhood of the fault site.

In the following, the most important steps of the proposed method will be described in more detail.



defect sensitive areas: a1, a2, a3, a4

overlap 1: a1 & a2 & a3 (→ max. connectivity fault)
2: a1 & a3 & a4 (→ max. connectivity fault)
3: a1 & a2
4: a1 & a3
5: a1 & a4
6: a2 & a3
7: a3 & a4

FIGURE 8 Partitioning the overlap areas.

4.2. Determination of Connectivity Faults

Procedures to compute the defect sensitive areas have been described in literature, e.g., [13]. This section deals with finding all possible connectivity faults and computing their critical areas for a fixed defect size. If the defect sensitive area of a fault primitive f_p does not overlap a defect sensitive area of any other fault primitive, the critical area of the connectivity fault $\{f_p\}$, $ca(\{f_p\})$, equals the defect sensitive area of f_p. If overlap exists, connectivity faults occur that include more than one primitive. Let f be a connectivity fault that includes the primitives $f_1,\ldots,f_k$. In order to determine its critical area $ca(f)$, we must determine the intersection of the defect sensitive areas $a_1,\ldots,a_k$ of $f_1,\ldots,f_k$ and subtract the critical areas of all connectivity faults f^Δ that include the same primitives $f_1,\ldots,f_k$ and at least one other fault primitive,

$$ca(f) = \bigcap_{i=1}^{k} a_i \Big\backslash \bigcup_{\substack{f^\Delta \supset f \\ f^\Delta \neq f}} ca(f^\Delta) \quad (2)$$

As the layout is partitioned into rectangular objects, the defect sensitive areas are always rectangular. But critical areas need not be rectangular (see Fig. 8), they even may be divided into several disjoint parts.

The critical area can be computed easily for the special class of connectivity faults given by the following definition.

DEFINITION 4 A connectivity fault f is a *maximal connectivity fault*, if $ca(f) \neq \emptyset$ and $ca(f^\Delta) = \emptyset$ holds for all $f^\Delta \supsetneq f$.

In the example of Figure 8, the maximal connectivity faults are $\{f_1, f_2, f_3\}$ and $\{f_1, f_3, f_4\}$. For a maximal connectivity fault the second term on the right hand side of (2) is empty, and the critical area is computed simply by intersecting the defect sensitive areas that correspond to the fault primitives of f.

After that, the critical areas can be determined for connectivity faults that include all the primitives of a maximal connectivity fault except one, in the next step all except two, and so on. Figure 9 shows an algorithm that implements the described approach.

```
Procedure DETERMINE_CONNECTIVITY_FAULTS
/*  input:  set of fault primitives and their defect sensitive areas */
/*  output: list of connectivity faults and their critical areas     */

determine the maximal connectivity faults;
L := list of all maximal connectivity faults ordered according
     to decreasing number of included fault primitives;
L^nz := Ø;  /* list of connectivity faults with nonzero critical area */

for all f∈L
  ca(f) := ⋂_{f_i∈f} a_i ; /* critical area of maximal connectivity faults */

repeat
  { f := first element of L;
    remove f from L and append f to L^nz;
    if f includes more than one fault primitive
      for all f_p∈f:
        { f^Δ := f \ {f_p};
          if f^Δ∉L          /* if fault f^Δ not yet considered */
            { ca(f^Δ) := ⋂_{f_i∈f^Δ} a_i \ ⋃_{f̃⊃f^Δ, f̃≠f^Δ, f̃∈L^nz} ca(f̃);
              if ca(f^Δ) ≠ Ø
                insert f^Δ into L according to the number of
                included primitives;
            }
          else
            { ca(f^Δ) := ca(f^Δ) \ ca(f); /* adjust critical area */
              if ca(f^Δ) = Ø /* remove fault that cannot occur */
                remove f^Δ from L;
            }
        }
  } until L = Ø;

/* L^nz contains all connectivity faults with nonzero critical area */
```

FIGURE 9 Determination of possible connectivity faults and their critical areas.

To determine the maximal connectivity faults, McCreigth's algorithm [20] can be applied reporting all pairs of intersecting defect sensitive areas. These correspond to the pairs of fault primitives occuring together in some connectivity faults. Then the maximal faults are determined using an algorithm which originally has been developed for the reduction of flow-tables [21]. For clarity, the procedure DETERMINE_CONNECTIVITY_FAULTS is described operating explicitly on 2-dimensional areas. If the critical areas are needed only to determine fault probabilities, it is sufficient to compute just their size, and the calculations can be simplified.

From the analysis of McCreigth's algorithm, it can be concluded that with n fault primitives at most $O(n^2)$ different connectivity faults with nonzero critical area are possible. Using this fact, it can be shown that both the time complexity and the space complexity of the algorithm are polynomial in n. The degrees of the polynomials depend on the details of the implementation.

4.3. Translation to Net List Faults

The connectivity faults are translated to net list faults using the connectivity graph. A fault primitive of type "undesigned connection" between two layout objects adds an edge between the corresponding vertices in the connectivity graph G. A fault primitive of type "disconnection" between two layout objects removes the edge that connects the corresponding vertices of G. And a fault primitive that disrupts one layout object replaces the corresponding vertex by two new vertices that are not connected by an edge (see Fig. 10). Connectivity faults result in a combination of such changes in the connectivity graph.

For the modified connectivity graph, the connected components (i.e., the nets resulting after fault injection) are extracted and compared to the fault-free nets. The differences determine the net list fault. Figure 11 outlines a procedure that translates an arbitrary connectivity fault to the fault model of Section 3. Generally, a connectivity fault affects only a small number of layout objects. As the layout objects have been marked with the net they belong to, the affected nets can be determined and only these nets have to be considered.

In order to compute the probability that a specific net list fault F occurs, the probabilities $P(f)$ of all the connectivity faults f that lead to this net list fault F are simply added. This is the advantage of uncorrelated faults in our approach.

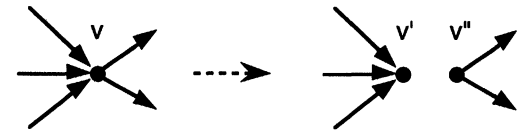


FIGURE 10 Disruption of the layout object represented by vertex v.

```
Procedure TRANSLATE_FAULT
/*  input:  • connectivity graph G = (V, E), vertices labeled   */
/*            with net number                                   */
/*          • connectivity fault f                              */
/*  output: • net list fault (BF, UF, or CF)                    */

BF := Ø;  UF := Ø;  CF := Ø;
modifiy G according to the fault primitives of f;
determine the set CC of connected components of G;

for each m∈CC
 { Nm := set of net labels that occur at the vertices of m;
   if |Nm| ≥ 2                  /* if several nets have been merged */
     BF := BF ∪ {Nm};                          /* bridging fault */
   for all n∈Nm                 /* for all nets (partially) included */
                                /* in connected component m      */
     { Vn := set of all vertices of V that are labeled with n;
       {m1, m2, ..., mr} := node sets of those components of CC
                            that contain vertices labeled with n;
                                  /* Vn ⊂ m1 ∪ m2 ... ∪ mr */
       if r > 1                   /* if net n has been partitioned */
         { Π̃(n) := {m1 ∩ Vn, m2 ∩ Vn, ..., mr ∩ Vn};
           translate Π̃(n) to Π(n) at electrical level;
           /* correspondence between electrical nodes and   */
           /* layout objects is known from circuit extraction */
           UF := UF ∪ {Π(n)};                    /* break fault */
         }
     }
 }

if (BF ≠ Ø and UF ≠ Ø)
  CF := (BF, UF);                              /* compound fault */
```

FIGURE 11 Translation of a connectivity fault to a net list fault.

The circuit described by the faulty net list can be simulated in order to evaluate its behavior at the electrical level. Some of the net list faults lead to voltages different from the fault-free case, others increase the quiescent power supply current, cause longer delay times, or introduce sequential behavior (e.g., stuck-open faults). Faults that lead to the same malfunction and hence require the same conditions to be satisfied for detection can be collapsed. Automatic test pattern generators can determine test patterns for voltage tests (logic tests) and I_{DDQ} tests, and pattern sequences to test sequential faults [22], [23], [24], [25].

5. EXPERIMENTAL RESULTS

The presented fault extraction method has been implemented in the software tool REFLEX (**re**alistic **f**ault **ex**traction). As an example, the OCTTOOLS standard cell library [26] was analyzed, which consists of 50 circuits with up to 33 transistors. The fabrication process is characterized by a 1 μm minimum feature size, two metal layers and one polysilicon layer. 10 defect mechanisms (for missing or extra material in each relevant layer) have to be considered. Realistic data for the defect densities of different defect mechanisms were obtained from a current fabrication line. The cells were analyzed for defects with sizes ranging from 1.0 to 10.0 μm in steps of 1.0 μm. The defect sizes were assumed to obey the $1/x^3$ distribution [27]. So defects larger than 10.0 μm occur with very low probability and can be neglected.

In order to obtain a measure for the expected yield of a certain cell, the grade has been defined as the reciprocal value of the probability that at least one of the extracted faults occurs:

$$\text{grade} := \frac{1}{1 - \prod_{\text{extracted faults } F} (1 - P(F))}$$

The grade of a cell can be used to compare different realizations of the same circuit regarding yield.

Table I shows the results for the standard cells that required the largest CPU times on a SUN SPARC 10. The remaining cells were analyzed in less than 5.9 seconds. The number of transistors, nets, and layout objects is denoted by *#T*, *#N*, and *#O*, respectively. The number of extracted bridging, break, and compound faults is marked by *#BF*, *#UF*, and *#CF*, respectively.

Altogether, 4973 faults were extracted from the cells of the library. Due to the precise description of faults, the number of extracted net list faults is much higher than the number of stuck-at faults. However, this number can be reduced using fault simulation. Net list faults that result in the same faulty behavior can be combined, and their probabilities can be added.

A classification of all the extracted faults from the cell library was performed. In order to get a

TABLE I Analysis of the OCTTOOLS standard cell library

name	function	area in μm^2	#T	#N	#O	#BF	#UF	#CF	CPU time in seconds	grade in 10^5
norf311	3 Input OR/NOR	1000	8	9	105	25	55	12	5.9	1.065
nanf311	3 Input NAND/AND	1872	18	9	105	23	49	12	6.7	0.901
aof2201	2.2 AND/OR Mux	2856	10	11	117	42	48	13	6.7	0.824
norf401	4 Input NOR	1480	8	10	96	33	59	12	6.8	1.085
xnof201	Exclusive NOR	2064	12	11	120	65	49	22	7.8	0.861
orf401	4 Input OR	2856	10	11	122	37	45	15	8.7	0.815
buff121	Tri-State Buffer	3776	10	8	142	25	41	16	9.4	0.873
xorf201	Exclusive OR	2160	12	11	124	53	50	20	10.1	0.781
nanf411	4 Input NAND/AND	2160	10	11	122	42	47	13	10.4	0.890
muxf201	Data Select	2064	12	12	127	56	48	20	11.2	0.722
delf011	Delay Cell	5040	10	9	162	34	54	17	13.6	0.928
aof3201	3.2 AND/OR Mux	5680	14	15	160	52	58	20	13.8	0.579
aof2301	2.3 AND/OR Mux	3648	14	15	171	61	66	21	21.6	0.558
aof4201	4.2 AND/OR Mux	8544	18	19	194	91	72	28	24.8	0.449
larf310	Clocked Latch	6600	18	14	213	77	77	29	31.2	0.426
dfnf311	D-FF with Q and QB	7200	24	18	243	103	84	47	71.9	0.354
faf001	Full Adder	11288	28	19	304	163	94	38	94.3	0.279
dfrf301	D-FF with Async. R and Q	15048	29	22	324	170	120	52	105.6	0.283
dfrf311	D-FF with Async. R.Q and QB	16632	31	23	358	174	133	53	137.4	0.270
dfbf311	D-FF with S.R.Q and QB	13944	33	24	415	231	135	61	287.8	0.195

realistic view of the relevance of different fault classes, the faults were weighted with their probability of occurrence. The resulting value is the probability that a fault which is actually observed in a real circuit belongs to a specific fault class. The classification yields the following results:

- Bridging faults play a dominant role, because "extra conductive material" defects occur 20...50 times more frequently than "missing conductive material" defects.
- V_{DD} or GND are involved in only half of the bridging faults. Only these faults can be described immediately as stuck-at faults.
- With probability 0.24, an actually observed bridging fault connects more than two nets. This clearly shows that multiple-net bridging faults cannot be neglected.
- About half of the break faults lead to a floating gate of one or more transistors. The other break faults can mostly be classified as single or multiple stuck-open faults.
- With probability 0.033, an actually observed break fault affects more than one net.
- Compound faults account for 0.2 % of all faults.

Table II shows the probability that an actually observed fault is caused by a defect with size less than 3, 4, 5, 6, or 10 μm. These data confirm that the upper bound of 10.0 μm used for the defect size interval is appropriate. With probability greater than 0.998, an actually observed fault is caused by a defect with size smaller than 10.0 μm.

In order to demonstrate the importance of considering multiple-net bridging faults, an AND/OR gate $y = x_0 \cdot x_1 \vee x_2$ was implemented on transistor level using the extracted netlists of OCTTOOLS standard cells. This combination of an AND- and an OR-gate was chosen because it is not a complex configuration rarely found in real circuits, but rather a simple combination that is frequently used in many designs. The fault-

TABLE II Probability of an observed fault being caused by a defect of a certain size

defect size interval	bridging fault	break and compound fault
0–3 μm	1.0%	0 %
0–4 μm	15.4 %	73.3 %
0–5 μm	82.4 %	83.9 %
0–6 μm	93.7 %	94.2 %
0–10 μm	99.8 %	99.9 %

free case and the following four bridging faults between the input lines were analyzed:

- bridging fault $f01 = \{\{x_0, x_1\}\}$ (short between x_0 and x_1)
- bridging fault $f02 = \{\{x_0, x_2\}\}$
- bridging fault $f12 = \{\{x_1, x_2\}\}$
- multiple-net bridging fault $f012 = \{\{x_0, x_1, x_2\}\}$

In order to model shorts accurately, their resistance values should be considered. According to the data reported in [2], most shorts have resistance values much lower than 500 Ω. Thus, the assumption of a small resistance, e.g., several hundred ohms, is realistic. In case of the example, any resistance $R < 675$ ohms leads to the same simulation results.

To assure equal driving strength of all input signal lines and to avoid any feedback of shorted lines to the driving source, the lines are buffered before the bridging occurs. Furthermore, a short may lead to a distorted output signal of a logic gate, which may be interpreted as "0" or "1" by a following gate. In order to get this "interpretation", the output line of the AND/OR gate is also buffered.

The results of a SPICE simulation [28] with all possible input patterns are depicted in Figure 12. The 2-net bridging faults are similar to wired-OR connections. The multiple-net bridging fault, however, leads to a more complex behavior.

An input pattern (x_2, x_1, x_0) detects a bridging fault if the output signals of the fault-free and the faulty case are different:

- $f01$ is tested by (0,0,1) and (0,1,0)
- $f02$ is tested by (0,0,1).
- $f12$ is tested by (0,1,0).
- $f012$ is tested by (1,0,0).

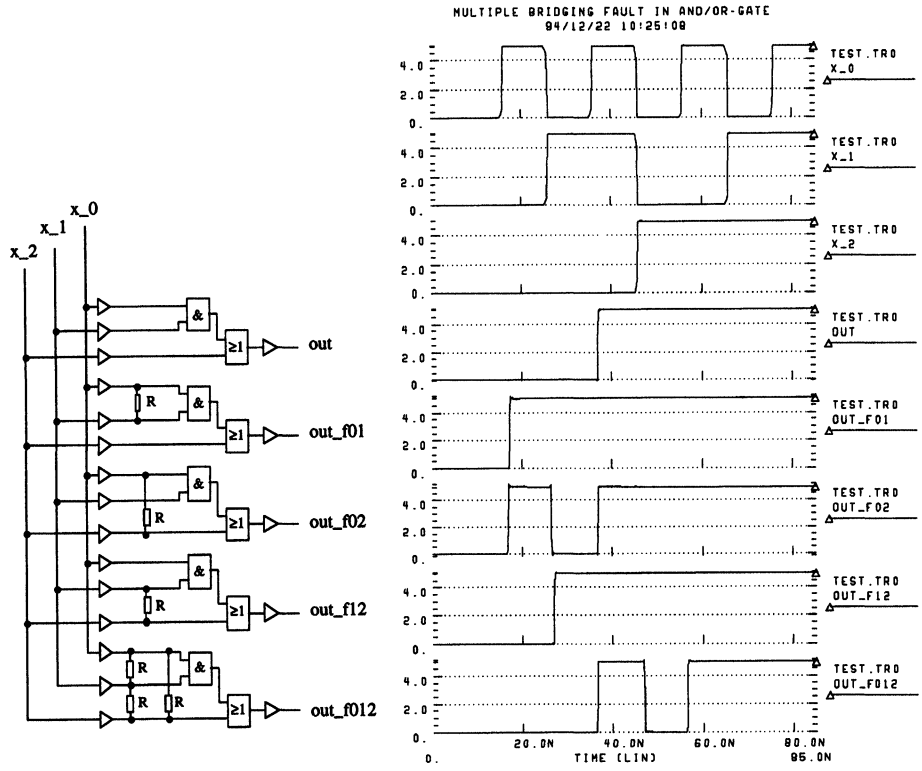


FIGURE 12 Results of a SPICE simulation with $R < 675$ ohms.

Even if all the test patterns for the 2-net faults are applied, the multiple-net fault is not detected. This clearly shows that for high quality tests it is not sufficient to consider only 2-net faults. As a consequence the probabilities of occurrence must also be determined for multiple-net faults, and our approach is justified.

6. PROCESSING LARGE LAYOUTS

The presented fault extraction method becomes even more efficient if specific characteristics of certain design styles are exploited. Gate-array designs, for example, have a regular structure concerning the placement and geometrical dimensions of their transistors. The layout consists of a large number of identical elements (so-called sites), which are connected by lines on one or more metal layers. Each gate-array cell has to be analyzed only once regarding both the interior of the cell and the border area to other cells, and these data are stored in the cell library.

Standard cell layouts can be partitioned into cells and routing channels. At first, these parts are analyzed separately. The information about the cells is collected from the cell library (containing the fault extraction data that is computed just once). The faults within the routing channels have to be determined specifically for the considered design. Routing channels often require a remarkable amount of layout area, and the signal lines inside a routing channel are typically much longer and more likely to be affected by a bridging or break fault than nodes internal to a logic gate. The fault extraction procedure has been adapted to the simple layout structures of routing channels [29]. The analysis is very efficient because no transistors have to be considered.

Similar as in gate-arrays, there may be defects which cross the border of two adjacent standard cells or the border of a standard cell and a routing channel. Those defects may cause faults which are not included in the fault list if standard cells or routing channels are analyzed separately. In order to handle these faults, we define a specific border area the size of which depends on the largest considered defect size (see Fig. 13). Then the fault extraction procedure is applied to the layout objects that partly or completely lie in the border area. In this step, only those faults have to be determined which involve layout objects of at least two different cells or one cell and a routing channel.

The proposed cell-based partitioning approach combines very well with hierachical test pattern generation tools that for each cell use local test vectors stored in the library and then apply propagation and justification procedures to determine global tests [23], [24], [25], [30]. Since the number of primary inputs of a standard cell usually is small, local test vectors can easily be obtained even for complex multiple-net faults inside the cell. The faults are injected one by one, and the faulty cell is simulated using all possible input patterns. When the response of the faulty cell differs from the fault-free response a test pattern has been found. Including the bridging faults that connect more than two nets and the break faults that affect more than one net, as proposed in this paper, increases the number of different faults only by a factor of about 2. This is a consequence of the $1/x^3$ distribution of defect sizes. Thus, the consideration of multiple-net faults causes only a moderate increase in CPU times required for fault simulation and test pattern generation.

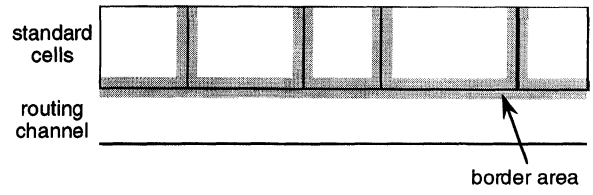


FIGURE 13 Border area in standard cell layouts.

7. CONCLUSIONS

Test pattern generation and fault coverage evaluation require a fault set that accurately describes the physical failures. In contrast to previous methods of inductive fault analysis, which considered only bridging faults between two nets and break faults splitting one net into two parts, this paper also considers multiple-net bridging faults, multiple break faults, and compound faults. The faults are modelled such that each defect can cause at most one fault. Hence, faults occur statistically independently if it is assumed that defects are statistically independent.

The presented fault extraction method analyzes the circuit at the layout level and in a bottom-up fashion determines all possible changes in the circuit structure at the electrical level. So all bridging, break, and compound faults that can actually occur are determined in a uniform way. As the faults are not correlated, their probability of occurrence can be computed very efficiently. The results obtained by the tool REFLEX show that bridging faults connecting more than two nets account for a significant portion of all faults.

In order to handle large circuit layouts, a partitioning approach is advantageous. Small standardized parts of the layout, e.g., cells, can be analyzed independently of each other. In a subsequent step, the border areas between these parts are considered.

The extraction of realistic faults and their probabilities provide a variety of useful applications in the area of design and test optimization. Fault probabilities can help to evaluate standard cells with respect to their defect sensitivity and detect particularly susceptible layout areas. It is possible to develop a set of design rules that consider defect sensitivity and support defect robust layout design. Furthermore, test sets can be ordered such that the tests detecting the most likely faults are applied first and test application time is reduced [9].

Acknowledgements

The authors wish to thank the anonymous referees for their constructive comments and suggestions.

References

[1] Eldred, R. D. (1959). "Test routines based on symbolic logical statements for combinational logic nets", *Journal of the ACM*, **6**(1), 33–36.

[2] Acken, J. M. (1988). "Deriving Accurate Fault Models", *Ph D Thesis*, Stanford University.

[3] Ferguson, F. J. and Larrabee, T. (1922). "Some Future Directions in Fault Modeling and Test Pattern Generation Research", *Technical Report UCSC-CRL-91-24*, University of California (Santa Cruz), Computer Engineering Department.

[4] Shen, J. P., Maly, W. and Ferguson, F. J. (1985). "Inductive Fault Analysis of MOS Integrated Circuits", *IEEE Design* and *Test*, **2**(6), 13–26.

[5] Walker, D. M. H. (1987). *Yield Simulation for Integrated Circuits*, Boston: Kluwer Academic Publishers.

[6] Stapper, C. H. (1989). "Fault-simulation programs for integrated-circuit yield estimations", *IBM Journal of Research and Development*, **33**(6), 647–652.

[7] Kuo, S.-Y. (1986). "YOR: A Yield-Optimizing Routing Algorithm by Minimizing Critical Areas and Vias", *IEEE Transactions on CAD*, **12**(9), 1303–1311.

[8] Teixeira, J. P., Goncalves, F. M. and Sousa, J. J. T. (1991). "Layout-Driven Testability Enhancement", *Proc. European Test Conference*, pp. 101–109.

[9] Spiegel, G. and Stroele, A. P. (1993). "Optimization of Deterministic Test Sets Using an Estimation of Product Quality", *Proc. Asian Test Symposium*, pp. 119–124.

[10] Ferguson, F. J. and Shen, J. P. (1988). "Extraction and Simulation of Realistic CMOS Faults using Inductive Fault Analysis", *Proc. International Test Conference*, pp. 475–484.

[11] Jacomet, M. (1989). "FANTESTIC: Towards a Powerful Fault Analysis and Test Pattern Generator for Integrated Circuits", *Proc. International Test Conference*, pp. 633–642.

[12] Stapper, C. H. (1983). "Modeling of Integrated Circuit Defect Sensitivities", *IBM Journal of Research and Development*, **27**(6), 549–557.

[13] Nigh, P. and Maly, W. (1989). "Layout-Driven Test Generation", *Proc. International Conference on CAD*, pp. 154–157.

[14] Jee, A. and Ferguson, F. J. (1993). "CARAFE: An Inductive Fault Analysis Tool for CMOS VLSI Circuits", *Proc. VLSI Test Symposium*, pp. 92–98.

[15] Corsi, F., Martino, S., Marzocca, C., Tangorra, R., Baroni, C. and Buraschi, M. (1992). "Critical Areas for Finite Length Conductors", *Microelectronics and Reliability*, **32**(11), 1539–1544.

[16] Xue, H., Di, Ch. and Jess, J. A. G. (1993). "A Net-Oriented Method for Realistic Fault Analysis", *Proc. International Conference on CAD*, pp. 78–83.

[17] Xue, H., Di, Ch. and Jess, J. A. G. (1994). "Probability Analysis for CMOS Floating Gate Faults", *Proc. European Design and Test Conference*, pp. 443–448.
[18] Stern, O. and Wunderlich, H. -J. (1994). "Simulation Results of an Efficient Defect Analysis Procedure", *Proc. International Test Conference*, pp. 729–738.
[19] Ousterhout, J. K. (1984). "Corner Stitching: A Data-Structuring Technique for VLSI Layout Tools", *IEEE Transactions on CAD*, **3**(1), 87–100.
[20] Ullman, J. D. (1984). *Computational Aspects of VLSI*, Rockville: Computer Science Press.
[21] Unger, S. H. (1969). *Asynchronous Sequential Switching Circuits*, New York: Wiley.
[22] Lee, H. K. and Ha, D. S. (1990). "SOPRANO: An Efficient Automatic Test Pattern Generator for Stuck-Open Faults in CMOS Combinational Circuits", *Proc. 27th ACM/IEEE Design Automation Conference*, pp. 660–666.
[23] Chess, B., Freitas, A., Ferguson, F. J. and Larabee, T. (1994). "Testing CMOS Logic Gates for Realistic Shorts", *Proc. International Test Conference*, pp. 395–402.
[24] Dalpasso, M., Favalli, M. and Olivo, P. (1995). "Test Pattern Generation for IDDQ: Increasing Test Quality", *Proc. VLSI Test Symposium*, pp. 304–309.
[25] Mahlstedt, U., Alt, J. and Heinitz, M. (1995). "CURRENT: A Test Generation System for IDDQ Testing", *Proc. VLSI Test Symposium*, pp. 317–323.
[26] *OCTTOOLS-5.2 User's Guide*, Department of Electrical Engineering and Computer Science, University of California, Berkeley, 1993.
[27] Stapper, C. H. (1984). "Modeling of defects in integrated circuits photolithographic patterns", *IBM Journal of Research and Development*, **28**(4), 461–475.
[28] Nagel, L. W. (1975). "SPICE2: A Computer Program to Simulate Semiconductor Circuits and Systems", *Memo ERL-M520*, University of California, Berkeley.
[29] Spiegel, G. (1994). "Fault Probabilities in Routing Channels of VLSI Standard Cell Designs", *Proc. VLSI Test Symposium*, pp. 340–347.
[30] Hansen, M. C. and Hayes, J. P. (1995). "High-Level Test Generation using Physically-Induced Faults", *Proc. VLSI Test Symposium*, pp. 20–28.

Authors' Biographies

Gerald Spiegel received a diploma degree in Electrical Engineering and Dr.-Ing. (Ph.D.) in Computer Science from the University of Karlsruhe, Germany. Currently, he is project manager at Digitaltest GmbH in Stutensee, Germany. There, he is responsible for the software development in the area of PCB testing and quality management. He is a member of the IEEE.

Albrecht P. Stroele received a diploma degree in Electrical Engineering from the University of Darmstadt, Germany, and Dr. rer. nat (Ph.D.) in Computer Science from the University of Karlsruhe, Germany. After several years with Siemens where he was involved in image processing and computer design, he joined the Institute of Computer Design and Fault Tolerance, University of Karlsruhe. His current research interests include fault modeling, CAD for testability, built-in self-test techniques, signature analysis, and cellular automata. He is a member of IEEE and GI (Gesellschaft fuer Informatik).