# A Unified Approach to the Extraction of Realistic Multiple Bridging and Break Faults

Gerald Spiegel, Albrecht P. Stroele

University of Karlsruhe
Institute of Computer Design and Fault Tolerance (Prof. Dr.-Ing. D. Schmid)
P.O. Box 69 80, D-76128 Karlsruhe, Germany

## Abstract

*The presented fault model uniquely describes all structural changes in the transistor net list that can be caused by spot defects, including faults that connect more than two nets and faults that break a net into more than two parts. The developed analysis method extracts the complete set of realistic faults from the layout and for each fault computes the probability of occurrence.*

## 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 too large to consider all of them explicitly. Fault models 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 real physical failures. 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 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. The growing density of integration in CMOS technology, however, increases the importance of bridging and break faults [1, 2], 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* [3] 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. [4, 5]), physical design for testability [6], and test optimization [7].

Several methods of inductive fault analysis have been developed in the past few years [3,8,9,10,11,12,13,14]. 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.)

Multiple-net bridging faults are not necessarily detected by tests for 2-net bridging faults. 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 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.

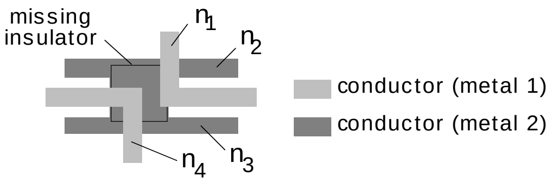

**Fig. 1: Correlation between faults**

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 circuit structure 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 a circuit layout with Manhattan geometry. The probability of occurrence is determined for each fault using a novel method for fast critical area calculation.

The next section deals with the representation of a circuit and its faults on layout and on transistor level. Section 3 introduces the improved realistic fault model. Section 4 describes the fault extraction method. Experimental results are discussed in section 5, and the conclusions are given in section 6.

## 2 Circuit and fault representation

As a bottom-up process, fault extraction starts at the layout level. We partition the geometric layout structures into rectangular regions. 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 objects are modelled by a *connectivity graph* G = (V,E). Its vertices V represent the 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 $C = \{c_1,...,c_q\}$, a set of transistors $T=\{t_1,...,t_r\}$, and a set of nets $N=\{n_1,...,n_s\}$. 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. 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 maximal connected component of the connectivity graph G.

Defects occur due to deviations and irregularities in the manufacturing process. In this paper, we consider *spot defects* with a size which is comparable to the minimum layout feature size. 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. 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 thereby 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.

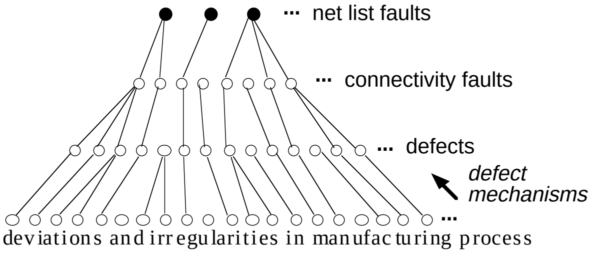

**Fig. 2: Defects, connectivity faults, and net list faults**

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 [10]). 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 used here model 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 $BF = \{N_1,..., N_v\}$ be a set of pairwise disjoint subsets of nets, $N_j \subset N$, j = 1, 2, ..., v, with at 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.

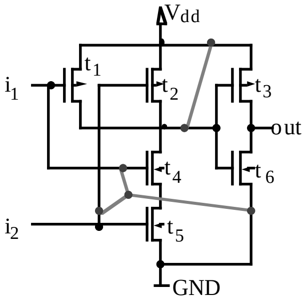

**Fig. 3: Bridging fault in a CMOS AND gate**

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

If 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 $UF = \{\Pi_1(n_1), ..., \Pi_w(n_w)\}$ be a set of partitions for the nets $n_i \in N$ with $|\Pi_i(n_i)| \geq 2$ for i = 1, 2, ..., w. UF is a *break fault* if a single spot defect can cause each net $n_i \in \{n_1, ..., n_w\}$ to be divided into two or more disconnected subnets according to $\Pi_i(n_i)$. All other nets are preserved.

Example: Figure 4 shows the 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 $UF = \{\Pi_1(n_2), \Pi_2(n_5)\}$ with $\Pi_1(n_2) = \{\{i_2, g(t_5)\}, \{g(t_2)\}\}$ and $\Pi_2(n_5) = \{\{d(t_1), s(t_4)\}, \{d(t_2)\}, \{g(t_3), g(t_6)\}\}$.

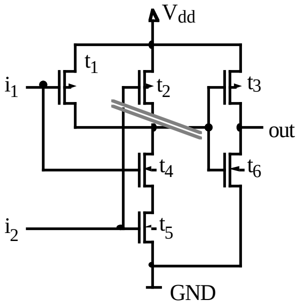

**Fig. 4: Break fault in a CMOS AND gate**

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.

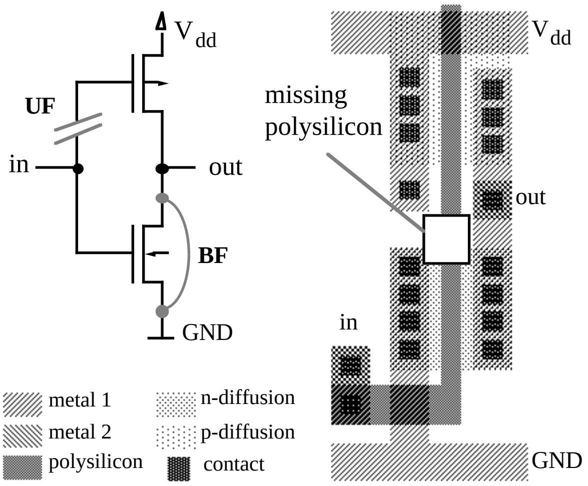

**Fig. 5: Compound fault in a CMOS inverter**

**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 net list at the electrical level. Moreover, all faults are uncorrelated, they occur statistically independent if the underlying defects occur statistically independent.

## 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, and a 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

and some special cases that depend on the applied technology, e.g. disruptions 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).

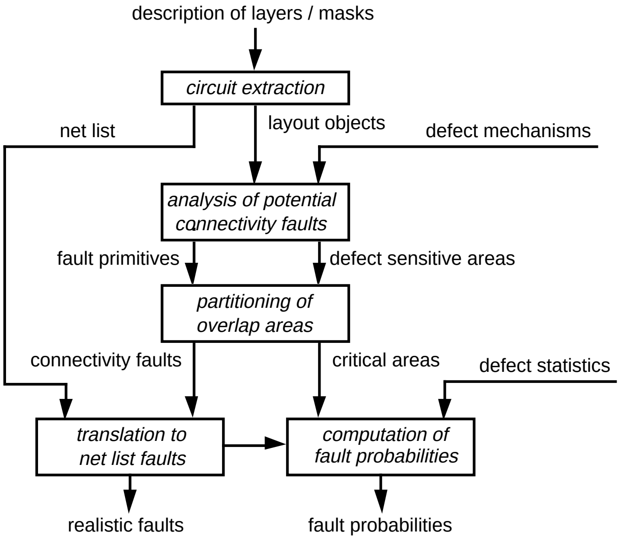

**Fig. 6: Fault extraction**

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, $\mathbf{f} = \{f_1,...,f_k\}$. This concept of a connectivity fault as a non-empty 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**, 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.

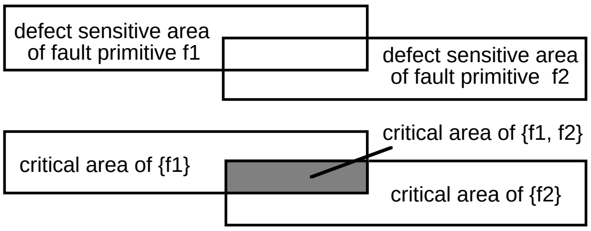
defect sensitive area of fault primitive f1
defect sensitive area of fault primitive f2
critical area of {f1, f2}
critical area of {f1}
critical area of {f2}

**Fig. 7: Defect sensitive area of fault primitives and critical area of connectivity faults**

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 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,

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

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 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.

## 4.2 Determination of connectivity faults

Procedures to compute the defect sensitive areas have been described in literature, e.g. [10]. 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 $f_1,...,f_k$. In order to determine its critical area ca(**f**), we must determine the intersection of the defect sensitive areas $a_1,...,a_k$ of $f_1,...,f_k$ and subtract the critical areas of all connectivity faults $\mathbf{f}^\Delta$ that include the same primitives $f_1,...,f_k$ and at least one other fault primitive,

$$ca(\mathbf{f}) = \bigcap_{i=1}^{k} a_i \setminus \bigcup_{\mathbf{f}^\Delta \supsetneq \mathbf{f}} ca(\mathbf{f}^\Delta) \qquad (2)$$

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

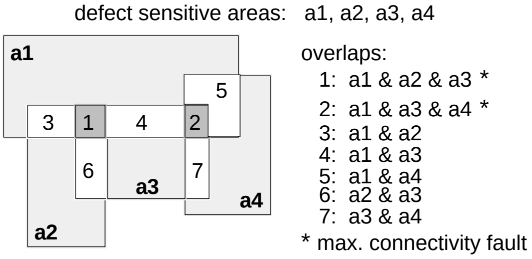

**Fig. 8: Partitioning the overlap areas**

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**) ≠ Ø and ca($\mathbf{f}^\Delta$) = Ø holds for all $\mathbf{f}^\Delta \supsetneq \mathbf{f}$.

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.

To determine the maximal connectivity faults, McCreigth's algorithm [15] can be applied reporting all pairs of intersecting defect sensitive areas. These correspond to the fault primitive pairs occuring together in some connectivity faults. For clarity, the procedure **GET_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.

```
Procedure   GET_CONNECTIVITY_FAULTS
/*  input:     set of fault primitives and defect sensitive  areas
    output:     list of connectivity faults and critical areas */
determine the maximal connectivity faults;
L   :=  list of all max. connectivity faults ordered acc. to
        decreasing number of included fault primitives;
$L^{nz}$  := Ø;
/*  determine critical area of max. connectivity faults  */
for all $\mathbf{f} \in L$:      $ca(\mathbf{f}) := \bigcap_{f_i \in \mathbf{f}} a_i$ ;
repeat
{   f  :=  first element of L;
  /*  $L^{nz}$: list of conn. faults with nonzero critical area  */
  remove f from L and append f to $L^{nz}$;
  if f includes more than one fault primitive then
    for all $f_p \in \mathbf{f}$:
        {  $\mathbf{f}^\Delta := \mathbf{f} \setminus \{f_p\}$;
           if $\mathbf{f}^\Delta \notin L$ then  /*  if $\mathbf{f}^\Delta$ not yet considered  */
              {   $ca(\mathbf{f}^\Delta) := \bigcap_{f_i \in \mathbf{f}^\Delta} a_i \setminus \bigcup_{\bar{\mathbf{f}} \supsetneq \mathbf{f}^\Delta,\, \bar{\mathbf{f}} \in L^{nz}} ca(\bar{\mathbf{f}})$;
                if  $ca(\mathbf{f}^\Delta) \neq Ø$ then insert $\mathbf{f}^\Delta$ into L acc. to
                  the number of included primitives;    }
           else     /*  adjust critical area  */
              {   $ca(\mathbf{f}^\Delta) := ca(\mathbf{f}^\Delta) \setminus ca(\mathbf{f})$;
                /*  remove fault that cannot occur  */
                if  $ca(\mathbf{f}^\Delta) = Ø$ then remove $\mathbf{f}^\Delta$ from L;      }
        }
}   until  L = Ø;
/* result:       list $L^{nz}$  */
```

**Fig. 9: Determination of possible connectivity faults and their critical areas**

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 figure 10). Connectivity faults result in a combination of such changes in the connectivity graph.

For the modified connectivity graph, the maximal connected components (i.e. the nets) are extracted and compared to the fault-free nets. The differences determine the net list fault. 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.

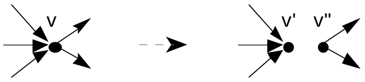

**Fig. 10: Disruption of the layout object represented by vertex v**

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.

## 5 Experimental results

The presented fault extraction method has been implemented in the software tool REFLEX. As an example, the OCTTOOLS standard cell library [16] 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 [17]. So defects larger than 10.0 µm occur with very low probability and can be neglected.

Table 1 shows the results for a choice of standard cells. The number of transistors and nets is denoted by #T and #N, respectively. The number of extracted bridging, break, and compound faults is marked by #BF, #UF, and #CF, respectively. The CPU time is for a SUN SPARC 10.

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.

| function | area ($\mu m^2$) | #T | #N | #BF | #UF | #CF | CPU time (sec.) |
|---|---|---|---|---|---|---|---|
| 3 Inp OR/NOR | 1000 | 8 | 9 | 25 | 55 | 12 | 5.9 |
| 4 Input NOR | 1480 | 8 | 10 | 33 | 59 | 12 | 6.8 |
| Exclus. NOR | 2064 | 12 | 11 | 65 | 49 | 22 | 7.8 |
| 4 Input OR | 2856 | 10 | 11 | 37 | 45 | 15 | 8.7 |
| Tri-St. Buffer | 3776 | 10 | 8 | 25 | 41 | 16 | 9.4 |
| Exclusive OR | 2160 | 12 | 11 | 53 | 50 | 20 | 10.1 |
| 4 NAND/AND | 2160 | 10 | 11 | 42 | 47 | 13 | 10.4 |
| Data Select | 2064 | 12 | 12 | 56 | 48 | 20 | 11.2 |
| Delay Cell | 5040 | 10 | 9 | 34 | 54 | 17 | 13.6 |
| AND/OR Mux | 8544 | 18 | 19 | 91 | 72 | 28 | 24.8 |
| Clocked Latch | 6600 | 18 | 14 | 77 | 77 | 29 | 31.2 |
| Full Adder | 11288 | 28 | 19 | 163 | 94 | 38 | 94.3 |
| D-FlipFlop 1 | 16632 | 31 | 23 | 174 | 133 | 53 | 137 |
| D-FlipFlop 2 | 13944 | 33 | 24 | 231 | 135 | 61 | 287 |

**Table 1: Analysis of the OCTTOOLS standard cell library**

After fault extraction, a classification of the extracted faults was performed. In order to get a more realistic view, the faults were weighted with their probability of occurrence. The classification yields the following results:

- Bridging faults play a dominant role, because "extra conductive material" defects occur 20-50 times more often 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.
- 24 % of the bridging faults connect 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.
- 3.3 % of the break faults affect more than one net.
- Compound faults account for 0.2 % of all faults.

The analysis also confirmed that the upper bound of 10.0 µm for the defect size interval is appropriate. With probability greater than 99.8 %, a fault is caused by a defect with size smaller than 10.0 µm.

## 6 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 show that bridging faults that connect more than two nets account for a significant portion of all faults.

## References

[1] J. M. Acken, ‘Deriving Accurate Fault Models’, PhD Thesis, Stanford Univ., 1988

[2] F. J. Ferguson, T. Larrabee, ‘Some Future Directions in Fault Modeling and Test Pattern Generation Research’, Techn. Rep. UCSC-CRL-91-24, Univ. of California (Santa Cruz), Comp. Eng. Dept., 1992

[3] F. J. Ferguson. J. P. Shen. W. Maly, ‘Inductive Fault Analysis of nMOS and CMOS Integrated Circuits’, Techn. Rep., Carnegie-Mellon Univ., Pittsburgh, 1985

[4] D. M. H. Walker, ‘Yield Simulation for Integrated Circuits’, Kluwer Acad. Pub., 1987

[5] C. H. Stapper, ‘Fault-simulation programs for integrated-circuit yield estimations’, IBM Journ. of Research and Development, Vol. 33,1989, pp.647-652

[6] J. P. Teixeira, F. M. Goncalves, J. J. T. Sousa, ‘Layout-Driven Testability Enhancement’, Europ. Test Conf., 1991, pp. 101-109

[7] G. Spiegel, A. P. Stroele, ‘Optimization of Deterministic Test Sets Using an Estimation of Product Quality’, Asian Test Symp., 1993, pp. 119-124

[8] M. Jacomet, ‘FANTESTIC: Towards a Powerful Fault Analysis and Test Pattern Generator for Integrated Circuits’, Int. Test Conf., 1989, pp. 633-642

[9] A. Jee, F. J. Ferguson, ‘CARAFE: An Inductive Fault Analysis Tool for CMOS VLSI Circuits’, VLSI Test Symp.,1993, pp.92-98

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

[11] H. Xue, Ch. Di, J. A. G. Jess, ‘A Net-Oriented Method for Realistic Fault Analysis’, Int. Conf. on CAD, 1993, pp. 78-83

[12] H. Xue, Ch. Di, J. A. G. Jess, ‘Probability Analysis for CMOS Floating Gate Faults’, Europ. Design and Test Conf., 1994, pp. 443-448

[13] G. Spiegel, ‘Fault Probabilities in Routing Channels of VLSI Standard Cell Designs’, VLSI Test Symp., 1994, pp. 340-347

[14] O. Stern, H.-J. Wunderlich, ‘Simulation Results of an Efficient Defect Analysis Procedure’, Int. Test Conf., 1994, pp. 729-738

[15] J. D. Ullman, ‘Computational Aspects of VLSI’, Computer Science Press, Rockville MD, 1984

[16] OCTTOOLS-5.2 User's Guide, Department of Electrical Engineering and Computer Science, Univ. of California, Berkeley, May 1993

[17] C. H. Stapper, ‘Modeling of defects in integrated circuits photolithographic patterns’, IBM Journ. of Research and Development,Vol. 28, 1984, pp.461-475