An Interactive Yield Estimator as a VLSI CAD tool

Israel A. Wagner
IBM Israel - Science and Technology
Matam, Haifa 31905, Israel

Israel Koren
Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, MA 01003

Abstract

The yield of a VLSI chip depends, among other factors, on the sensitivity of the chip to defects occurring during the fabrication process. To predict this sensitivity, one usually needs to compute the so-called **critical area** *(A_c) which reflects how many and how large the defects must be in order to result in a circuit failure. The main computational problem in yield estimation is to calculate A_c efficiently for complicated, irregular layouts. A novel approach is suggested for this problem that results in an algorithm to solve it efficiently. This algorithm is compared to other yield-prediction methods, which use either the Monte-Carlo approach (VLASIC) or a deterministic approach (SCA), and is shown to be faster. It also has the advantage that it can graphically show a detailed 'defect sensitivity map' that can assist a physical designer in improving the yield of his/her layout.*

1 Introduction

Following [2],[4],[5], the yield of a chip, Y, is computed as $Y = \prod_{i=1}^{m} Y_i$ where Y_i is the yield associated with the ith step of the manufacturing process. For convenience, the subscript will be omitted and Y will be referred to as the yield of a single processing step. Using the negative binomial distribution, the yield of a single processing step is modeled as

$$Y = \left(1 + \frac{dA_c}{\alpha}\right)^{-\alpha} \tag{1}$$

where d denotes the average number of defects per unit of area, α the clustering parameter and A_c the critical area, defined by

$$A_c = \int_0^\infty A(r)D(r)dr \tag{2}$$

$A(r)$ is the area in which the center of a defect of radius r must fall in order to cause a circuit failure. For a given circuit layout C it is defined as

$$A(r) = \int\int_{(x,y)\in C} \delta(x,y,r)dxdy \tag{3}$$

where

$$\delta(x,y,r) = \begin{cases} 1 & \text{a defect of radius } r \text{ at } (x,y) \text{ causes a circuit failure} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

0-8186-3502-9/93 $3.00 © 1993 IEEE

$D(r)$ is the density function of the defect size. Experimental data on defects in many wafers lead to the following formula ([4],[5],[7]):

$$D(r) = \begin{cases} cr^q/r_0^{q+1} & 0 \le r \le r_0 \\ cr_0^{p-1}/r^p & r_0 \le r \le \infty \end{cases} \tag{5}$$

where p and q are real numbers (typically $p \approx 3, q \approx 1$ [7]), and c is given by $c = (q+1)(p-1)/(q+p)$ [1].

The problem in yield estimation is to calculate A_c for a complicated, irregular layout. Existing yield-prediction algorithms are based on one of the following methods:

- *Deterministic Approach:* Compute $A(r)$ for several values of r by either the scan-line method ([8],[9]) or shape-expansion ([3],[6]). Equation (2) is then used to approximate A_c, and equation (1) is used to predict the yield.
- *Monte-Carlo Approach* (e.g.,[10],[11]): Draw a large number of defects, with their radii distributed according to $D(r)$, check for each defect if it causes a fault, then divide the number of defects causing circuit failures by the total number of defects.

Algorithms based on the deterministic approach are usually faster, while the Monte-Carlo approach yields more precise results, but needs substantially more CPU time. Hence, neither method is suitable as a precise interactive design tool. In this paper a new analytic approach is suggested that leads to a faster algorithm for computing A_c.

The rest of the paper is organized as follows: Section 2 includes the suggested analytic approach, Section 3 describes the algorithm and in Section 4 some numerical and graphic results are presented.

2 Critical Area Analysis - A New Approach

2.1 Computing the Critical Area

In general, $A(r)$ is an area measure, and hence can be expressed as an integral, for a given cell C, as was shown in Equations (3) and (4) above.

Let us denote by u an ordered set of coordinates (x, y). Thus

$$A_c = \int_{r=0}^{\infty} D(r) \left(\int_{u \in C} \delta(u, r) du \right) dr \tag{6}$$

Note that u is a two-dimensional point; hence du is an infinitely small area unit. The two integrations are independent, so their order can be changed:

$$A_c = \int_{u \in C} \left(\int_{r=0}^{\infty} \delta(u, r) D(r) dr \right) du \tag{7}$$

Let us denote by $S(u)$ the internal integral. In fact, $S(u)$ is the defect-sensitivity at point u, i.e., the probability of a circuit fault caused by a defect at point u, and is given by

$$S(u) = \int_{r=0}^{\infty} \delta(u, r) D(r) dr \tag{8}$$

We then have

$$A_c = \int_{u \in C} S(u) du \tag{9}$$

Let us assume that $\delta(u,r)$ is monotonic in r; that is, if a defect of size r at point u results in a circuit fault, then a defect of size $r' > r$ at the same point will also result in a circuit fault. Further, let $r_c(u)$ denote the critical radius at point u, i.e., the minimal size of a defect causing a circuit fault at point u.

$$r_c(u) = \min_r\{\delta(u,r) = 1\}$$

This will simplify the formula for $S(u)$ resulting in

$$S(u) = \int_{r=r_c(u)}^{\infty} D(r)dr \tag{10}$$

By now the problem of calculating the critical area A_c has been reduced to that of calculating the critical radius at each point of the layout, since the integral in equation (10) can be pre-calculated for several values of r_c. In practice, if a simple formula like (5) is used, then its integral can be represented in a closed form.

In order to be practical, let us consider the area of the layout as a set of grid points, each of which may be either empty or belong to a shape in the layout. The grid resolution is γ, which has to be small enough to cover all significant shape variations. If our layout has width W and height H, then there are WH/γ^2 points in it. The layout is described as a set of shapes $L = \{P_1, P_2, \ldots, P_k\}$ where the P's are sets of points, and k is the number of shapes in the layer under consideration. In what follows 'points' has the meaning of 'grid points'.

Based on the above analysis one can employ a naive algorithm for calculating $r_c(p)$, as depicted in Figure 1. The problem with **ALG1** is its high complexity. There are WH/γ^2 points, and for each point we need to scan all the other points (in the worst case, when there are only 2 small polygons near the edge of the area). Hence, the complexity of running **ALG1** on the whole cell is $O\left(WH/\gamma^2\right)^2$ For example, if the layout is $100\mu \times 100\mu$, and we set $\gamma = 0.1\mu$, then we need $O(10^{12})$ computation steps.

2.2 Computing $r_c(u)$ Efficiently - A Ring Theorem

Let us first discuss the 'short-circuit' type faults. In this context, $r_c(u)$ is the minimal radius of a circle around u that intersects more than one shape. A defect with such a radius will cause a short-circuit.

We denote by $|u - u'|$ the distance between the points u and u'. (Although we use the Euclidean distance in this paper, the theory holds for other measures as long as the triangle inequality holds.) Also, we denote by $r_i(u)$ the distance from u to the ith nearest shape of the layout; clearly $r_c(u) = r_2(u)$.

The corollary of the following theorem states that the critical radii of two points cannot differ by more than the distance between these points.

Theorem 1 *: Assume that u and u' are two points of the layout. Then*

$$\forall i \in \{1,2\}: \ |r_i(u) - r_i(u')| \leq |u - u'|$$

Corollary 1 *: Any polygon that may affect $r_c(u)$ should intersect either the ring*

$$\{v \mid r_1(u') - |u - u'| \leq |v - u| \leq r_1(u') + |u - u'|\}$$

or the ring

$$\{v \mid r_2(u') - |u - u'| \leq |v - u| \leq r_2(u') + |u - u'|\}$$

The proofs of Theorem 1 and Corollary 1 are omitted for the sake of brevity.

Using Theorem 1, one can devise a better algorithm for $r_c(u)$, that makes use of the previously calculated values $r_1(u')$ and $r_2(u')$, where u' is a neighbor of u on the grid. (See Figure 2).

In **ALG2** we perform an exhaustive search for only one point. Then, for all the other points we need only scan a ring whose radius cannot exceed the edge of the cell, and whose width is 2γ. Hence, the worst-case complexity of running **ALG2** on the whole cell is only $O\left(WH/\gamma^2\right)^{1.5}$ which, in the example above, is 1000 times faster than **ALG1**. A more sophisticated analysis leads to an even better bound of $O\left(WH\overline{r_c}/\gamma^3\right)$ where $\overline{r_c}$ is the average critical radius of the layout. It is interesting to note that the complexity is inversely proportional to the layout density; as the number of shapes grows, $\overline{r_c}$ decreases, as does the complexity of **ALG2**.

Using equations (9) and (10) above, and the procedure **ALG2** to calculate $r_c(x,y)$, the sensitivity and critical area for an arbitrary cell can be approximated as depicted in Figure 3. In order to get a good approximation for A_c, one must choose a γ such that (1) γ is small enough so that we will not miss any significant variation in the critical radius between grid-points, and (2) the number of test-points, given by WH/γ^2, is not too large, in order to keep the algorithm efficient in time and space.

The following theorem shows the linkage between γ and the maximal error:

Theorem 2 : *Denote by A_c the critical area of the layout, and by A'_c the critical area as computed by ALG3. Then, for a typical distribution of defect size, the relative error satisfies*

$$\left|\frac{A'_c - A_c}{A_c}\right| \leq \frac{0.943\gamma}{r_g}$$

where r_g is the minimum ground rule for the layer under consideration.

For example, if $r_g = 1.0\mu m$ and γ is selected to be $0.01\mu m$, then the relative error is bounded by 0.943%. In other words, to have the error bounded by ϵ we should set

$$\gamma \leq \frac{\epsilon \cdot r_g}{0.943} \leq 1.061 \cdot \epsilon \cdot r_g$$

In fact, our implementation of **YMAP** calculates a better bound on its error using the actual values of critical radii.

3 Implementation and Results

YMAP was implemented in the *C* programming language, and run on an RS/6000 IBM workstation. It has been integrated into the IBMS design system [12]. Figure 4 shows examples of metal polygons with the corresponding defect-sensitivity map for short-circuit faults.

3.1 Usage of the YMAP Algorithm

The values calculated by the algorithm can be used in two ways:

1. Yield-Meter: for a given layout cell L, compute the critical area, then use equation (1) to evaluate the yield of the cell.

2. Yield-Pointer: Draw a 'topographic map' of L, such that the intensity of color at point u is proportional to $S(u)$ as defined in equation (8) (See Figure 4).

If $Y(L)$ is too small, one can use the sensitivity map to find how the yield can be improved.

3.2 Comparison With Other Yield Estimators

We compared the YMAP program to two yield simulators: VLASIC ([10],[11]) and SCA ([8],[9]). Simple layouts were considered for which analytic formulas of the critical area could be derived. For each such layout, we ran YMAP, VLASIC and SCA for several time limits, and then compared their relative results. The time limit was set by either the number of defects (VLASIC), the step size of the defect radius (SCA), or the grid size (YMAP). The results are 'relative' in the sense that we take one case as having unit critical area, and then normalize the other cases accordingly. This was done for two reasons. First, it is important for an interactive yield simulator tool to be able to compare several layout configurations for their predicted yield. Second, when two programs are being compared, their different 'numerical noise' may be filtered out by normalization.

A Test Case: Short-Circuit Faults in Two Parallel Lines

For the case of short-circuit faults between two parallel lines (see Figure 5) of width w each and spacing s, it can be shown that the critical area is

$$A_c = H \cdot \left(\frac{2}{s} - \frac{1}{w+s} - \frac{w+s/2}{R_{max}^2} \right) \tag{11}$$

where $(2w+s)$ and H are the horizontal and vertical dimensions of the cell, respectively, and R_{max} is the maximum possible radius of a defect for the given layout. In Table 1 we compare the convergence speed of the three algorithms, that is, how fast each algorithm converges to the precise ratio of $A_c/A_c(par_1)$. The numbers in the table are normalized according to par_1. The advantage of using YMAP is demonstrated in Figure 5, where the convergence rates of the programs are compared graphically.

A Test Case: Short Circuit Faults for Two Corners

In this case the layout was made of two squares in opposite corners of the area under test. The convergence rates are quite similar to those achieved for parallel lines. The detailed results are omitted for the sake of brevity.

3.3 Discussion

All three algorithms (VLASIC, SCA and YMAP) eventually reach the correct results. It seems that the Monte-Carlo procedure used by VLASIC requires many more iterations for the results to be precise. On the other hand, SCA and YMAP rapidly converge to their final result (YMAP's speed is somewhat better), and are deterministic, so a single run is sufficient.

Probably, the VLASIC approach is more suitable for a whole chip analysis when a long, batch type execution is acceptable. YMAP, on the other hand, can be more useful as a part of a cell/macro CAD system, when a physical designer needs a fast and precise yield prediction of his/her layout.

If the chip is a memory unit built from a repeated basic block, then YMAP constitutes a very efficient tool. Another advantage of YMAP is that it provides a bound on its error, so the user has a better knowledge about the precision of the results.

Acknowledgment: The work of the second co-author was supported in part by NSF under contract MIP-9305912.

References

[1] A.V. Ferris-Prabhu, "Defect Size Variations and Their Effect on the Critical Area of VLSI Devices," *IEEE Journal of Solid State Circuits,* vol. sc-20, pp. 878-880, 1985.

[2] A.V. Ferris-Prabhu, *Introduction to Semiconductor Device Yield Modeling,* Artech House, 1992.

[3] S. Gandemer, B.C. Tremintin, J.J. Charlot, "Critical Area and Critical Levels Calculation in IC Yield Modeling," *IEEE Transactions on Electronic Devices,* vol. 35, no. 2, pp. 158-166, February 1988.

[4] I. Koren, "The Effect of Scaling on the Yield of VLSI Circuits," in *Yield Modelling and Defect Tolerance in VLSI Circuits,* W.R. Moore, W. Maly and A. Strojwas (Eds.), pp. 91-99, Adam Hilger Ltd., 1988.

[5] I. Koren, A.D. Singh, "Fault Tolerance in VLSI Circuits," *IEEE Computer magazine,* pp. 73-83, July 1990.

[6] M. Rivier, "Random Yield Simulation Applied to Physical Circuit Design," in *Yield Modelling and Defect Tolerance in VLSI Circuits,* W.R. Moore, W. Maly and A. Strojwas (Eds.), pp. 111-120, Adam Hilger Ltd., 1988.

[7] C.H. Stapper, "Modelling of Defects in Integrated Circuits Photolithographic Patterns," *IBM J. Res. Dev.,* vol. 28, no. 4, pp. 461-475, 1984

[8] J. Pineda de Gyvez and C. Di, "IC Defect Sensitivity for Footprint Type Spot Defects," *IEEE Transactions on CAD,* vol. 11, no. 5, pp. 638-658, May 1992.

[9] J. Pineda de Gyvez, *SCA – a defect-sensitivity evaluation system for IC layout,* Texas A&M University.

[10] D. M. H. Walker, *VLASIC System User Manual, Release 1.3,* CMU, August 1990.

[11] H. Walker, S.W. Director, "VLASIC: A Catastrophic Fault Yield Simulator for Integrated Circuits," *IEEE Trans. on CAD,* vol. 5, no. 4, pp. 541-556, Oct. 1986.

[12] *IBMS - Integrated Book and Macro System,* IBM Science and Technology, Haifa, Israel.

Function r $_c(u)$:real;
var r: real; begin
    $r := 0$;
    repeat
        $r := r + \gamma$;
        scan the points in the r-vicinity of u
    until you find two points, u_1, u_2 that belong to two different shapes;
    $r_c(u) := \max\{|u - u_1|, |u - u_2|\}$
end;

Figure 1: ALG1 – Naively computing $r_c(u)$.

Function $\mathbf{r_c}(u)$:real;
var d: real;
begin
$d := |u - u'|$
repeat
scan the points in the two rings:
R1:$\{v \mid r_1(u') - d \leq |v - u| \leq r_1(u') + d\}$
R2:$\{v \mid r_2(u') - d \leq |v - u| \leq r_2(u') + d\}$
until you find two points, u_1, u_2 that belong to two different shapes and are nearest to u;
$r_1(u) := \min\{|u - u_1|, |u - u_2|\}$;
$r_2(u) := \max\{|u - u_1|, |u - u_2|\}$;
$r_c(u) := r_2(u)$;
end;

Figure 2: ALG2 – Efficiently computing $r_c(u)$ using $r_1(u'), r_2(u')$.

Procedure YMAP(L : *layout*; ϵ_{max}, r_g : *real*);

Input: * A layout L with width W and height H.
* ϵ_{max} - A bound on the error in A_c.
* r_g - The minimal ground rule for the layer under test.
Output: * The defect sensitivity $S(x, y)$ for each grid-point $(x, y) \in L$.
* A_c = The critical area of L.

---

var A, γ, r_c: real;
i, j, m, n: integer;
begin
$\gamma := 1.061 \cdot \epsilon_{max} \cdot r_g$; /* this number results from Theorem 2 */
$A := 0$, $m := \lceil W/\gamma \rceil$, $n := \lceil H/\gamma \rceil$
for $i := 1$ to m, $j := 1$ to n do
$r_c := \mathbf{r_c}(i\gamma, j\gamma)$;
$S(i\gamma, j\gamma) := \int_{t=r_c}^{\infty} D(t)dt$;
$A := A + \gamma^2 S(i\gamma, j\gamma)$; /* γ^2 is an area element for piecewise integration*/
end;

Figure 3: YMAP – Computing the defect-sensitivity $S(\cdot)$ and the critical area A_c.

VLASIC	CPU time (sec)	0.44	1.77	3.7	9.38	15.81	82.84	160.01
	Avg. Error	70.45%	22.66%	20.56%	17.11%	34.08%	4.19%	1.68%
YMAP	CPU time (sec)	0.31	1.53	8.43	25.08	57.52	79.17	161.11
	Avg. Error	11.65%	5.44%	3.49%	2.54%	1.98%	1.60%	1.46%
SCA	CPU time (sec)	1.24	4.17	9.65	15.11	20.81	91.73	162.71
	Avg. Error	58.76%	16.88%	3.82%	1.86%	0.93%	0.21%	0.16%

Table 1: Comparing the average errors in $A_c/A_c(par1)$ of YMAP, VLASIC and SCA for parallel lines.

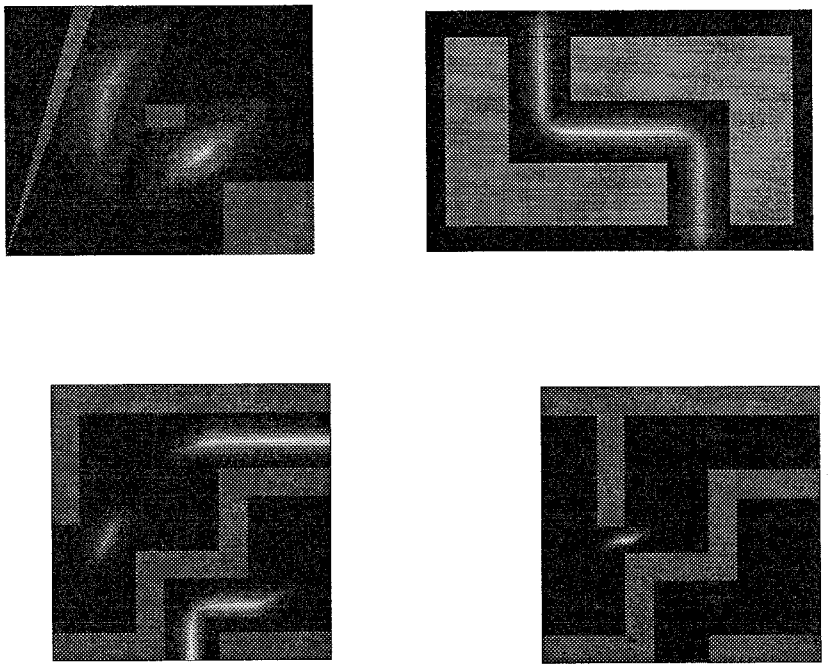

Figure 1: Examples of sensitivity maps produced by YMAP.

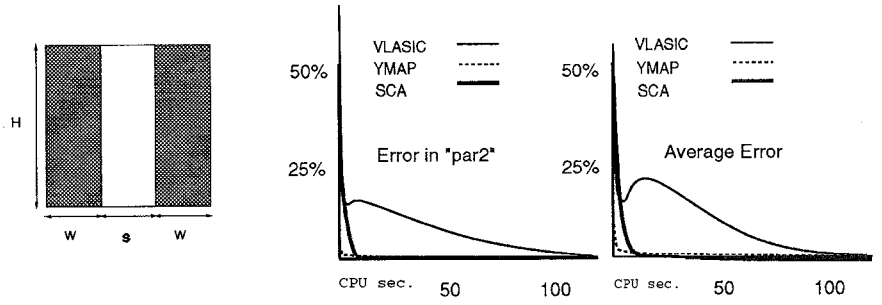

Figure 2: Cell with parallel lines and convergence rates for VLASIC, SCA and YMAP.