Critical Area Computation via Voronoi Diagrams

Evanthia Papadopoulou and D. T. Lee, *Fellow, IEEE*

***Abstract*— In this paper, we present a new approach for computing the critical area for shorts in a circuit layout. The critical area calculation is the main computational problem in very large scale integration yield prediction. The method is based on the concept of Voronoi diagrams and computes the critical area for shorts (for all possible defect radii, assuming square defects) accurately in $O(n \log n)$ time, where n is the size of the input. The method is presented for rectilinear layouts and layouts containing edges of slope ± 1. As a byproduct, we briefly sketch how to speed up the grid method of Wagner and Koren [18].**

***Index Terms*— Critical area, defects, shorts, very large scale integration (VLSI) layout, Voronoi diagrams, yield prediction.**

I. Introduction

THE critical area of a very large scale integration (VLSI) layout is a measure that reflects the sensitivity of the layout to defects occurring during the manufacturing process and it is widely used to predict the yield of a VLSI chip. Yield prediction is essential in today's VLSI manufacturing due to the growing need to control cost. Models for yield estimation are based on the concept of critical area which represents the main computational problem in the analysis of yield loss due to spot defects during fabrication. Spot defects are caused by particles such as dust and other contaminants in materials and equipment and are classified into two types: "extra material" defects causing shorts between different conducting regions and "missing material" defects causing open circuits. Extra material defects are the ones that appear most frequently in a typical manufacturing process and are the main reason for yield loss. For more information on yield estimation and spot defects, see, for example, [4], [5], [8], [11], [12], [14], and [16]–[18].

Quoting from [18], the yield of a chip, denoted by Y, is computed as $Y = \Pi_{i=1}^{m} Y_i$, where Y_i is the yield associated with the ith step of the manufacturing process. The yield of a single processing step is modeled as

$$Y_i = (1 + dA_c/\alpha)^{-\alpha}$$

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

For a circuit layout C, the critical area is defined as

$$A_c = \int_0^\infty A(r)D(r)\,dr$$

where $A(r)$ denotes the area in which the center of a defect of radius r must fall in order to cause circuit failure and $D(r)$ is the density function of the defect size. The defect density function has been estimated as follows [5], [8], [17], [18]:

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

where p, q are real numbers (typically, $p = 3, q = 1$), $c = (q+1)(p-1)/(q+p)$, and r_0 is some minimum optically resolvable size.

Existing methods for yield prediction and critical area computation can be classified into the following types:

1) *Geometric Methods:* Compute $A(r)$ for several different values of r independently. Use the results to approximate A_c. The methods to compute $A(r)$ are usually based on *shape-expansion* followed by *shape-intersection* (see e.g., [6]) and have a quadratic flavor for medium or large defect sizes. For rectilinear layouts there is a more efficient scan-line method which can compute $A(r)$ in multiple layers [14].
2) *Virtual Artwork Approach:* Build a virtual artwork having the same statistical features as the nominal itegrated circuit (IC) layout. The virtual artwork is arranged in a form allowing easy calculation of the critical area as a function of the defect radius [10]. Accuracy is limited by differences in the details of the nominal and virtual artworks.
3) *Monte Carlo Approach:* Draw a large number of defects with their radii distributed according to $D(r)$, check for each defect if it causes a fault, and divide the number of defects causing faults by the total number of defects [20].
4) *Grid Approach:* Assume an integer grid over the layout. Compute the critical radius (the radius of the smallest defect causing a fault at this point) for every grid point [18]. The method works in $O(I^{1.5})$ time, where I is the number of grid points. The accuracy depends on the density of the grid.

In this paper, we present a new approach for computing the critical area for shorts in a single layer. The method is based on the concept of *Voronoi diagrams* and can accurately compute

Manuscript received June 10, 1998; revised October 3, 1998. The work of D. T. Lee was supported in part by the Office of Naval Research under Grant N00014-97-1-0514 and in part by the National Science Foundation (NSF) under Grant CCR-9731638. This paper was recommended by Associate Editor M. Wong.

E. Papadopoulou is with the IBM T.J. Watson Research Center, Yorktown Heights, NY 10598 USA (e-mail: evanthia@watson.ibm.com).

D. T. Lee is with the Department of Electrical and Computer Engineering, Northwestern University, Evanston, IL 60208 USA.

Publisher Item Identifier S 0278-0070(99)02324-6.

the total critical area A_c for square defects in $O(n \log n)$ time, where n is the size of the input. In this paper, we focus on rectilinear layouts but we also consider layouts containing edges of slope ± 1. This is the type of layout appearing most often in practice. Our method is extendible to layouts with shapes in arbitrary orientations (see [13]). To the best of our knowledge, this is the first low-polynomial algorithm to compute critical area for shorts accurately in irregular layouts.

This paper is organized as follows. In Section II, we show the connection between the L_∞ metric and square defects. In Section III, we review Voronoi diagrams of polygons in the L_∞ metric and in Section IV, we show the relevance of the second-order Voronoi diagram to critical area computation. In Section V, we show that the critical area for shorts can be computed analytically once the second-order Voronoi diagram is available. In Section VI, we give a simple plane sweep algorithm to compute the Voronoi diagram of rectangles. Finally, in Section VII, we show that the implicit mechanical construction of the second-order Voronoi diagram using a grid can speed up the method of [18] to $O(I)$, where I is the number of grid points.

II. Square Defects and the L_∞ Metric

In the critical area literature, defects have usually been modeled as circles due to the common use of Euclidean geometry. To simplify the computation, in this paper, a spot defect of size r is modeled as a square of side $2r$ (i.e., a square of radius r). In reality, spot defects have any kind of shape; thus, modeling defects as squares is as good as the circle model. Moreover, in existing methods, critical area is approximated in a way that would not make much difference if the defects were assumed squares or circles. (In [14] defects are also modeled as squares.) Modeling defects as squares corresponds to computing critical area in the L_∞ metric instead of the Euclidean. In L_∞, the distance between two points $p = (x_p, y_p)$ and $q = (x_q, y_q)$ is the maximum of the horizontal and the vertical distance between p and q, i.e., $d(p, q) = \max\{|x_p - x_q|, |y_p - y_q|\}$. Intuitively, the L_∞ distance is the size of the smallest square touching p and q. In contrast, the Euclidean distance corresponds to the size of the smallest circle touching p and q. Note that the "unit circle" in the L_∞ metric is a square of side two (i.e., a square of radius 1). The L_∞ distance between a point p and a line l is $d(p, l) = \min\{d(p, q), \forall q \in l\}$.

Since the L_∞ distance between any two points is less or equal to the Euclidean distance between the points, the critical area in the L_∞ metric is always an upper bound to the Euclidean critical area. A formal bound is given in Lemma 2, Section V. In the following, unless otherwise noted, we always imply the L_∞ distance and thus we skip the term for brevity when there is no confusion.

III. The Voronoi Diagram of Polygons

In L_∞, the bisector of two elements (lines or points) is the locus of points at equal L_∞ distance from the two elements. It can be regarded as the locus of points corresponding to centers of squares touching the two elements. Figs. 1 and 2 illustrate the L_∞ bisector of two points and two lines, respectively. Note that in case of two points along the same horizontal or vertical line the bisector consists of a line segment and two unbounded regions (shaded regions in Fig. 1). Without creating any significant difference we assign one region to one point and consider only the outermost boundary of the bisecting region as the bisector (thick lines in Fig. 1).

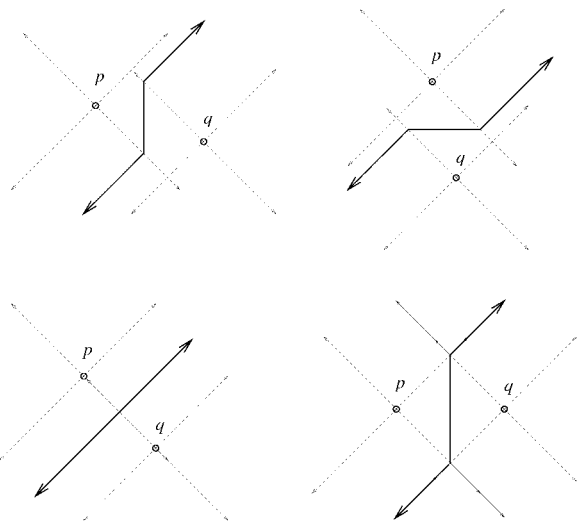


Fig. 1. The L_∞ bisector of two points.

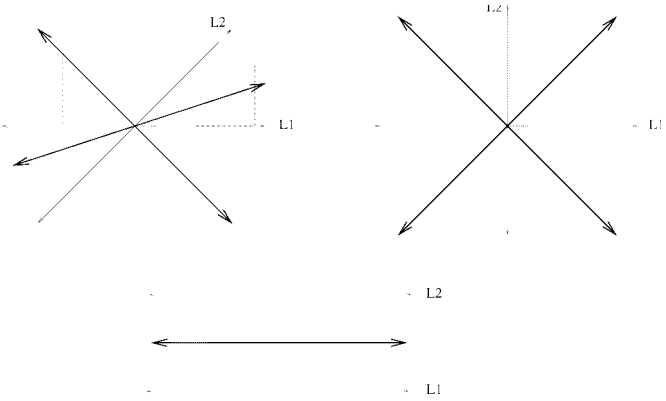


Fig. 2. The L_∞ bisector of two lines.

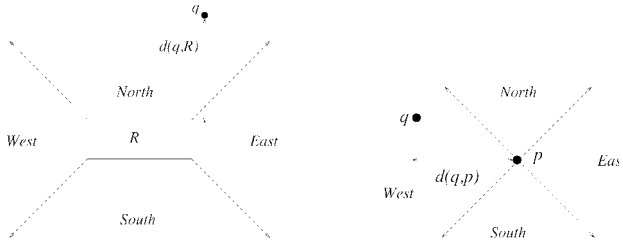


Fig. 3. The 45° rays emanating from the corners of R and from p partition the plane into four quadrants.

Consider a rectangle R and the four 45° rays[1] emanating away from the vertices of R (see Fig. 3). They partition the plane into four quadrants. For any point q in the north or

[1] A 45° ray is a ray of slope $+1$ or -1.

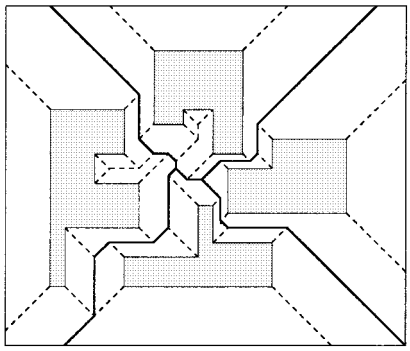


Fig. 4. The L_∞ Voronoi diagram of four polygons.

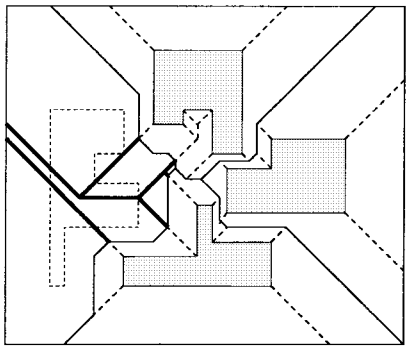


Fig. 5. The second-order Voronoi diagram in $\mathrm{reg}(P)$.

south (respectively, east or west) quadrant, the L_∞ distance to R simplifies to the vertical (respectively, horizontal) distance between q and the line through the respective horizontal (respectively, vertical) edge of R. The four 45° rays through the corners of R are regarded as bisectors between the edges of R. Similarly, for a point p; the four 45° rays emanating from p partition the plane into four quadrants where distance simplifies into vertical or horizontal (see Fig. 3). The rays can be regarded as bisectors between the vertical and horizontal distance from p. In this paper, we focus on the rectilinear case and, thus, for simplicity, we usually consider rectilinear shapes.

The Voronoi diagram of a set of disjoint polygons is a partition of the plane into regions, called *Voronoi cells*, each of which is associated with a polygon, called the *owner*, of the cell. The Voronoi cell of a polygon P, denoted as $\mathrm{reg}(P)$, is the locus of points closer to P than to any other polygon (see for example Fig. 4). The Voronoi cell of P is further partitioned into finer regions each of which is associated with an element (edge or vertex) of P. The Voronoi cell of an element $e \in P$ is the locus of points closer to e than to any other element. Clearly, $\mathrm{reg}(e) \subset \mathrm{reg}(P)$. The boundary that borders two Voronoi cells is called a *Voronoi edge*, and consists of portions of *bisectors* between the owners of the cells. The point where three or more Voronoi edges meet is called a *Voronoi vertex*. For more information on Voronoi diagrams in general, see [1], [2]. In the Euclidean metric the boundary of a Voronoi cell contains parabolic curves which make the computation difficult in practice. For this reason we perform computations in the L_∞ metric.

The L_∞ Voronoi diagram of a set of polygons is a skeleton of straight line segments with combinatorial complexity linear in the number of polygonal vertices [13]. In the special case of rectilinear polygons, the L_∞ Voronoi diagram is a particularly simple skeleton. It consists of line segments in only four orientations, vertical, horizontal, and lines of slope $(+1)$ and (-1) [13]. Figs. 4 and 8 illustrate the Voronoi diagram of a set of rectilinear polygons and a set of rectangles, respectively. Solid edges illustrate the Voronoi edges separating the cells of different polygons and dashed edges illustrate Voronoi edges induced by the edges of the same polygon. Note that a Voronoi vertex is the meeting point of at least three Voronoi edges (bisectors) and, thus, it is equidistant from at least three elements. An upper bound on the size of the L_∞ Voronoi diagram of a set of rectilinear polygons is $(2n-2)$, where n is the number of vertices [13].

The Voronoi diagram encodes nearest neighbor information for every point in the plane. For any point $t \in \mathrm{reg}(e)$, where e is an element of polygon P, t is closer to P than to any other polygon. Moreover, t is closer to e than to any other polygonal element (in the L_∞ sense). We say that polygon P, and in particular element e, is the *nearest neighbor* of t. Several algorithms are available in the literature for computing the Euclidean Voronoi diagram of line segments in time $O(n \log n)$ in expected or worst case (see [2] for a survey). Most of them could be modified to compute the simpler L_∞ diagram however, not all are simple to implement. In Section VI, we give a simple plane sweep algorithm to compute the L_∞ Voronoi diagram of a set of rectangles which can be easily generalized to rectilinear polygons. For a generalization of this algorithm to polygons in arbitrary orientations see [13].

IV. Voronoi Diagrams for Critical Area

Let us assume that the set of polygons C represents a layer of a VLSI layout. The *critical radius* of a point t is the radius of the smallest defect centered at t which overlaps with at least two different polygons (shapes) in different nets. The critical radius of t reflects the size of the smallest defect that would cause a short at point t. To determine the critical radius of a point t whose nearest polygon is P we need the second nearest polygon to t, say Q (assuming that P and Q belong to different nets). If w is the element of Q nearest to t, the critical radius of t is $d(w,t) = d(Q,t)$.

Consider now the partitioning of the plane obtained as follows: For every polygon $P \in C$, partition the interior of $\mathrm{reg}(P)$ by the Voronoi diagram of $C - P$. The subdivision derived in this manner is called the *second-order Voronoi diagram* of C and provides second nearest neighbor information. In Fig. 5, the thick lines illustrate the *second-order Voronoi diagram* restricted in the interior of $\mathrm{reg}(P)$ where P is shown in dotted lines. More formally, the second-order Voronoi region of an element $s \in C - P$ within the Voronoi cell of P is defined as $\mathrm{reg}_P(s) = \{x \mid d(s,x) \leq d(t,x), \forall t \in C - P\}$. Region $\mathrm{reg}_P(s)$ is *owned* by the unique pair (P,s). The critical radius of every point $x \in \mathrm{reg}_P(s)$ is the distance between x and s. Note that a square centered at x overlapping with s must also overlap with P (since P is closer to x than s) and, thus,

it causes a short. In other words, s is the element responsible for shorts within $\mathrm{reg}_P(s)$. Because s is the element inducing the critical radius for every point $x \in \mathrm{reg}_P(s)$, we drop P and say that s is the *owner* of $\mathrm{reg}_P(s)$.

This second-order Voronoi diagram of polygons is based on the concept of the *kth order Voronoi diagram* of point sites [9], which is defined as a planar subdivision such that each region belongs to a set H of k sites and is the locus of points closer to all sites in H than to any other site not in H. The size of the *second-order Voronoi diagram* cannot be more than twice the size of the ordinary Voronoi diagram, i.e., it is linear in the size of the input. To derive the second-order diagram within the Voronoi cell of each polygon P, we only need to consider the Voronoi neighbors of $\mathrm{reg}(P)$. In fact, as shown in Fig. 5, the second-order Voronoi edges in $\mathrm{reg}(P)$ are obtained by extending the 1st order Voronoi edges incident to the boundary of $\mathrm{reg}(P)$ in the interior of the cell. More details on how to compute the second-order diagram in the rectilinear case are given in Section VI. In the following, we assume that the second-order Voronoi diagram is available.

A detail worth pointing out in the computation of critical area for shorts is that a pair of distinct polygons may be part of the same net via a connection through a different layer. In this case, a defect overlapping with these polygons does not cause a short and, thus, should not be contributing to critical area. In the above approach it is easy to accommodate this point. Let P, Q be two disjoint polygons that belong to the same net. If the Voronoi cells of P and Q are not neighboring, then any defect overlapping P and Q must also overlap with some other polygon. If, on the other hand, the regions are neighboring, we can simply unite $\mathrm{reg}(P)$ and $\mathrm{reg}(Q)$ into $\mathrm{reg}(P \cup Q)$ by removing the adjacent Voronoi edges and treat the resulting region as any other when computing the second-order diagram.

V. Computing Critical Area

We have a layer in a circuit layout consisting of a collection of rectilinear polygons C. We assume that overlapping polygons have been unified into single shapes and, thus, we can assume that all polygons are disjoint. Moreover, we can assume that all polygons are *simple*, i.e., they do not contain holes. Note that holes are irrelevant to shorts and, thus, they can be ignored. The boundary of the layout is assumed to be a rectangle B. A defect overlapping a shape and the boundary is not considered to create a short. Our goal is to compute the critical area for shorts, i.e., to evaluate the integral $A_c = \int_0^\infty A(r)D(r)\,dr$, where the defect density function is given by (1). Using typical values for p, q and c we derive the widely used defect size distribution $D(r) = r_0^2/r^3$. (Since r_0 is typically smaller than the minimum feature size, we ignore $D(r)$ for $r < r_0$.) Recall that $A(r)$ denotes the area of the *critical region* for square defects of radius r. The critical region for radius r is the locus of points where if the center of a square defect of radius r is placed it causes a short, i.e., the defect overlaps with two polygons in different nets.

Let us assume that we are given the second-order L_∞ Voronoi diagram of C (with the modification described in Section IV to accommodate polygons of the same net). This

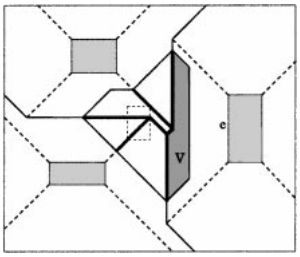


Fig. 6. Second-order Voronoi cell of vertical edge e.

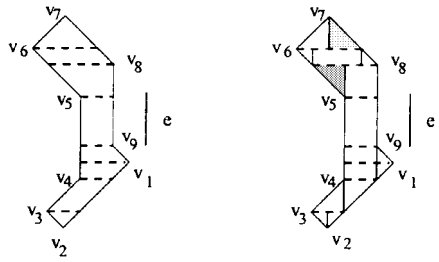


Fig. 7. The decomposition of V into trapezoids.

diagram essentially provides a partitioning of the plane into regions where critical area is easy to compute. The points of intersection of the boundary B with the Voronoi diagram are regarded as Voronoi vertices. The portion of the boundary between two consecutive vertices can be regarded as a Voronoi edge and is referred to as a *boundary edge*. Note that boundary edges are only bounding Voronoi cells and do not have regions of their own.

It is not hard to see that the total critical area of the layout is the sum of the critical areas within each (second-order) Voronoi cell. That is, $A_c = \sum_V A_c(V)$ for all (second-order) Voronoi cells V, where $A_c(V)$ denotes the critical area within V. Note that $A_c(V) = \int_0^\infty A(r, V)D(r)\,dr$, where $A(r, V)$ denotes the area of the critical region for defect radius r within V. Let us first concentrate on computing critical area within a single (second-order) Voronoi cell V having as owner an edge e. Let us assume without loss of generality that e is a vertical edge. Since the edges of V are bisectors induced by e they must be vertical or 45°. If V is incident to the boundary, then it may be bounded by a horizontal boundary edge which, however, can be ignored as it will be evident in the sequel. In Fig. 6, the shaded region illustrates a second-order Voronoi cell of the vertical edge e.

Consider a decomposition of V into trapezoids by drawing the horizontal lines emanating from the Voronoi vertices of V [see Fig. 7(a)]. Each trapezoid $\mathcal{T}$ can be further decomposed into triangles and at most one rectangle $\mathcal{R}$ by drawing the vertical lines through its vertices, see [Fig. 7(b)]. In case $\mathcal{T}$ is a slanted parallelogram, continue the decomposition recursively. Note that the hypotenuse of the triangles (if any) is portion of a 45° Voronoi edge. Note also that the (second-order) Voronoi cell of a horizontal edge e' would be decomposed into trapezoids by drawing vertical lines through its vertices. We distinguish between two kinds of triangles in the above decomposition depending on the relevant position with respect

to e of their vertical edge and opposite apex. If the apex lies between e and the vertical edge (comparing abscissae) the triangle is called *diverging*; otherwise, it is called *converging*. In Fig. 7, the lightly shaded triangle is diverging and the darker shaded one is converging. Given two vertices v_j and v_k, such that v_j is closer to e than v_k, let r_j, r_k denote the corresponding critical defect radii, i.e., $r_j = |x_e - x_j|$ and $r_k = |x_e - x_k|$, where x_j, x_k, and x_e denote the x-coordinates of v_j, v_k, and e, respectively.

Consider a rectangle $\mathcal{R}$, a diverging triangle T_{div}, and a converging triangle T_{cnv} in the above decomposition of $\mathcal{T}$. Using $D(r) = r_0^2/r^3$ and algebraic manipulation we derive the following formulas for the critical area within $\mathcal{R}, T_{\text{div}}$ and T_{cnv}.

Lemma 1: The critical area within a rectangle $\mathcal{R}$, a diverging triangle T_{div}, and a converging triangle T_{cnv} is given by the following formulas, using the "r_0^2/r^3" defect density distribution:

$$A_c(\mathcal{R}) = r_0^2/2(l/r_j - l/r_k) \tag{2}$$

$$A_c(T_{\text{div}}) = r_0^2/2(\ln(r_k/r_j) - l/r_k) \tag{3}$$

$$A_c(T_{\text{cnv}}) = r_0^2/2\big(l/r_j - \ln(r_k/r_j)\big) \tag{4}$$

where l is the size of the vertical side of $\mathcal{R}, T_{\text{div}}$ and T_{cnv}, and r_k, r_j, $r_k > r_j$, are the maximum and the minimum critical radius of their vertices.

Proof: Consider the rectangle $\mathcal{R}$ in the above decomposition. The area of the critical region within $\mathcal{R}$ for defect radius r, where $r_j \le r \le r_k$, is $A(r, \mathcal{R}) = l(r - r_j)$, where l is the length of vertical side of $\mathcal{R}$. For a defect radius $r \ge r_k$, the whole rectangle $\mathcal{R}$ is critical and, thus, the area is $A(r, \mathcal{R}) = l(r_k - r_j)$. Hence, the critical area within $\mathcal{R}$ is $A_c(\mathcal{R}) = r_0^2 l(\int_{r_j}^{r_k} (r - r_j)/r^3\, dr + \int_{r_k}^{\infty} (r_k - r_j)/r^3\, dr) = r_0^2/2(l/r_j - l/r_k)$.

Let T_{div} be a diverging triangle within the same decomposition. Let l, w be the length of the vertical and the horizontal edge of T_{div}, respectively. Since T_{div} is orthogonal and isosceles, $l = w = r_k - r_j$. The area of critical region within T_{div} for defect radius r where $r_j \le r \le r_k$, is $A(r, T_{\text{div}}) = 1/2 l_r w_r$, where $l_r = w_r = r - r_j$. For defect radius $r \ge r_k$, the whole triangle is critical and, thus, $A(r, T_{\text{div}}) = 1/2 lw$. Hence, the critical area within T_{div} is $A_c(T_{\text{div}}) = r_0^2/2(\int_{r_j}^{r_k} (r - r_j)^2/r^3\, dr + \int_{r_k}^{\infty} (r_k - r_j)^2/r^3\, dr) = r_0^2/2(\ln(r_k/r_j) - l/r_k)$.

For a converging triangle T_{cnv}, consider the diverging triangle T_{div}, obtained by flipping T_{cnv} around the hypotenuse. T_{cnv} and T_{div} form a rectangle $\mathcal{R}$. For any defect radius $r \ge r_j$, the area of critical region within T_{cnv} is $A(r, T_{\text{cnv}}) = A(r, R) - A(r, T_{\text{div}})$. Thus, $A_c(T_{\text{cnv}}) = \int_0^{\infty} (A(r, R) - A(r, T_{\text{div}}))D(r)\, dr = A_c(\mathcal{R}) - A_c(T_{\text{div}})$. The formula is derived by arithmetic substitution. □

We derive the critical area within V by adding up the critical areas within every rectangle and triangle in the above decomposition of V. Because of the summation, terms of the form l/r corresponding to internal decomposition edges cancel out. Similarly, for logarithmic terms involving endpoints of the decomposition other than Voronoi vertices. Let us classify the edges bounding V as *red* or *blue* as follows: A vertical edge is colored red (respectively blue) if the interior of V and the owner e lie on opposite sides (respectively, same side) of the edge. A 45° edge incident to a diverging triangle is colored red, and a 45° edge incident to a converging triangle is colored blue. A vertical boundary edge of V (if any) is clearly colored blue (recall that e is vertical). A horizontal boundary edge of V (if any) is equivalent to a horizontal decomposition edge and does not get involved in the formulas of Lemma 1, i.e., it receives no classification and is irrelevant to the critical area computation. In the formulas of Lemma 1, terms corresponding to red edges get added, while terms corresponding to blue edges get subtracted. The Voronoi edges bounding the cell of a horizontal edge are classified similarly.

To derive the total critical area for the layout, we add the critical areas within every (second-order) Voronoi cell. Every Voronoi edge bounds exactly two cells and receives the same coloring (red or blue) with respect to both cells. Boundary edges bound exactly one cell and their classification is blue or they are unclassified. We derive the following theorem.

Theorem 1: Given the second-order L_∞ Voronoi diagram of rectilinear polygons of a layer in a circuit layout C and assuming that defects are squares following the "r_0^2/r^3" defect density distribution, the critical area for shorts in that layer is given by the following formula:

$$A_c = r_0^2 \Bigg(\sum_{\text{red}\, e_i} l_i/r_i - \sum_{\text{blue}\, e_i} l_i/r_i + \sum_{\text{red}\, e_{45}} \ln(r_k/r_j) - \sum_{\text{blue}\, e_{45}} \ln(r_k/r_j) - 1/2 \sum_{\text{blue}\, b_i} (l_b/r_b) \Bigg)$$

where l_i and r_i denote the length and the critical radius of an orthogonal Voronoi edge, $r_k, r_j, r_k > r_j$ denote the maximum and the minimum critical radius of a 45° Voronoi edge, and l_b, r_b denote the length and the critical radius of a boundary edge. The first two summations are taken over all red and all blue orthogonal Voronoi edges, respectively. The third and forth summations are taken over all red and all blue 45° Voronoi edges, respectively. The last summation is taken over all blue boundary edges.

Thus, the critical area problem for shorts in a rectilinear layout is reduced to the second-order Voronoi diagram of polygons. The critical area can be written as a function of Voronoi edges and, thus, it can be computed analytically once the (second-order) Voronoi diagram is available. As will be shown in Section VII, this result also holds for layouts containing 45° edges; for a generalization to arbitrary shapes, see [13].

The following lemma gives a bound on the value of critical area in the L_∞ versus the Euclidean metric.

Lemma 2: Assuming that $D(r) = r_0^2/r^3$, the critical area in the L_∞ metric, A_c^∞, is bounded by the Euclidean critical area, A_c^e, with the following relation:

$$A_c^e \le A_c^\infty \le 2A_c^e.$$

Proof: As shown in [18], the critical area can be written as $A_c = \int_{u \in C} S(u)\, du$, where u denotes a point on layer C, $S(u) = \int_{r=r_c(u)}^{\infty} D(r)\, dr$, and $r_c(u)$ is the critical radius

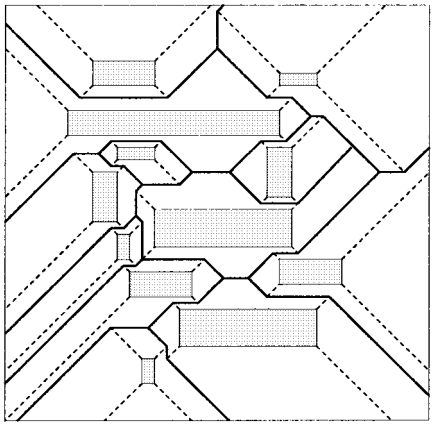


Fig. 8. The L_∞ Voronoi diagram of rectangles.

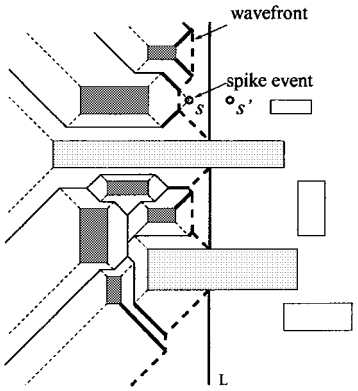


Fig. 9. Construction of the Voronoi diagram by plane sweep method.

of point u. For $D(r) = r_0^2/r^3$, $S(u) = r_0^2/(2r_c^2(u))$. Thus, $A_c = r_0^2/2 \int_{u \in C} 1/r_c^2(u)\, du$.

Let $r_e(u)$ and $r_\infty(u)$ denote the Euclidean and the L_∞ critical radius at point u, respectively. Let t and t' denote the points along the polygonal boundary that determine $r_e(u)$ and $r_\infty(u)$, respectively. In other words, $r_e(u) = d_e(u,t)$ and $r_\infty(u) = d_\infty(u,t')$, where d_e and d_∞ denote the Euclidean and the L_∞ distance, respectively. Let s and s' be the points along the polygonal boundary nearest to u in the Euclidean and the L_∞ metric, respectively. By the definition of critical radius, s and t (respectively, s' and t') must belong to different nets. Clearly, $r_\infty(u) \le \max\{d_\infty(u,t), d_\infty(u,s)\} \le \max\{d_e(u,t), d_e(u,s)\} = r_e(u)$.

Consider the square of radius $r_\infty(u)$ centered at u and its circumcircle $\mathcal{D}$. The radius of $\mathcal{D}$ is $\sqrt{2}r_\infty(u)$. Points s' and t' must clearly lie within $\mathcal{D}$. Then point t must also lie within $\mathcal{D}$, otherwise, $r_e(u)$ would be determined by s' or t', i.e., $r_e(u)$ could be reduced. Thus, $r_e(u) = d_e(u,t) \le \sqrt{2}r_\infty(u)$. Since $A_c^e = r_0^2/2 \int_{u \in C} 1/r_e^2(u)\, du$, $A_c^\infty = r_0^2/2 \int_{u \in C} 1/r_\infty^2(u)\, du$, and $\sqrt{2}r_\infty \ge r_e \ge r_\infty$, we derive that $A_c^e \le A_c^\infty \le 2A_c^e$. □

VI. A Sweep-Line Algorithm to Compute the L_∞ Voronoi Diagram of Rectilinear Polygons

In this section we give a plane-sweep algorithm to compute the L_∞ Voronoi diagram of a rectangular layout consisting of n rectangles (see Fig. 8). We consider rectangles instead of rectilinear polygons for simplicity of presentation; the same algorithm can be easily generalized to rectilinear layouts. The time complexity is $O(n \log n)$ and it is based on the wavefront paradigm introduced by Dehne and Klein [3] for the Voronoi diagram of points in *nice* metrics.

We shall apply a vertical sweepline $\mathcal{L}$ sweeping across the entire plane from left to right. The set of rectangles at any instant of the sweeping process will be partitioned into three subsets, S_l, S_m, and S_r, corresponding to those that lie totally to the left of $\mathcal{L}$, intersect $\mathcal{L}$, and lie totally to the right of $\mathcal{L}$, respectively. In Fig. 9, S_l, S_m, and S_r are shown darkly shaded, lightly shaded, and unshaded, respectively. The sweep line, which is cut by the rectangles in S_m, is decomposed into $|S_m| + 1$ sweepline segments, two of which are unbounded. Each such sweepline segment is considered to be the left edge of a *moving* rectangle.

We shall maintain a partial Voronoi diagram computed so far for the rectangles in S_l, the portion of the rectangles in S_m that lie to the left of $\mathcal{L}$, and the sweepline segments. At any instant of the sweeping process the boundary of the Voronoi cell associated with a single sweepline segment is called a *wavefront curve* and consists of bisectors induced by the sweepline segment. The collection of the wavefront curves for all sweepline segments is referred to as the *wavefront*. In Fig. 9, the wavefront is shown in dashed-lines. The Voronoi edges that have an endpoint common with the wavefront are called *spike bisectors* and are shown in solid and thicker lines in Fig. 9. As the sweepline moves to the right, the wavefront will shift to the right and so do the spike bisectors.

There are two major components associated with the plane-sweep algorithm: an *event list* and a *sweepline status*. The event list, implemented as a *priority queue* Q, consists of abscissae, called *priority values*, at which the sweepline status will change. The sweepline status, implemented as a *height-balanced* binary tree $\mathcal{T}$, implicitly maintains the wavefront. In particular, the sweepline status contains the spike bisectors and the north and south edges of the rectangles in S_m. The ordering follows the order of the wavefront.

To simulate the wavefront movement to the right as the plane sweep proceeds, we parameterize the endpoints of spike bisectors with respect to the abscissa of the sweep line t. In other words, we keep the abscissa of the endpoint of a spike bisector as a linear function of t. This function can be easily derived by the property that the endpoints must be equidistant (in L_∞) from the sweepline and the owners of the spike bisector. The abscissa of the endpoint of a north or south edge in $\mathcal{T}$ is given by t. At any instant t, the wavefront is the polygonal line connecting the endpoints of the elements of $\mathcal{T}$. Note that for any two endpoints u and v of consecutive elements in $\mathcal{T}$, segment $\overline{u,v}$ defines a bisector $B(e_b, s)$, where s is one of the $|S_m| + 1$ sweepline segments of $\mathcal{L}$ and e_b is the edge in $S_l \cup S_m$, which is the common owner of u and v.

In the event list we keep the *events* where the sweepline-status will change. The events are of two kinds: 1) events corresponding to vertical edges of rectangles referred to as *edge events* and 2) events corresponding to the intersection

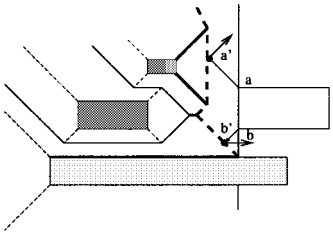


Fig. 10. A west edge event.

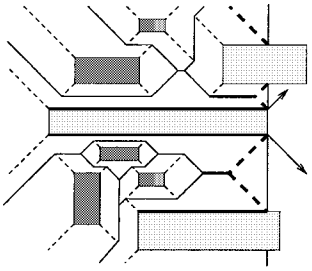


Fig. 11. An east edge event.

point of two neighboring spike bisectors referred to as *spike events*. The priority value of an edge event is the abscissa of the edge. A spike event corresponds to a potential Voronoi vertex that may or may not appear in the final Voronoi diagram. The priority value of a spike event s of abscissa x_s is $x_s + d(e_j, s)$, where e_j is the owner of s ($s \in \text{reg}(e_j)$). For example, in Fig. 9, the spike event associated with the potential Voronoi vertex s has as priority value the abscissa of the point marked as s'. That is, the spike event corresponding to s occurs when the sweepline reaches s'.

The priority value of an event corresponds to the time when the occurrence of the event affects the wavefront for the first time. This must be clear for the priority value of an edge event. The priority value of a spike event corresponds to the time when the wavefront reaches the corresponding intersection point for the first time. Thus, an unprocessed spike event lies always in front of the wavefront; it becomes part of the wavefront at time equal to its priority value (see also [3]). Edge events and spike events are the only reason for the wavefront to change its structure (see [3]). Thus, between any two events with consecutive priority values the structure of the wavefront remains the same.

Initially, the event list Q consists of *all* the abscissae of the west and east edges of the set of rectangles, and the sweepline status $\mathcal{T}$ is empty. We sweep $\mathcal{L}$ across the plane in ascending order of the abscissae, stopping at each event point of a certain priority value. When there are more than one abscissa that corresponds to west or east edge of a rectangle, we adopt the following tie-breaking convention: an east edge always precedes a west edge, and if the edges are of the same type, the one with the smaller ordinate will be processed first.

Let us now discuss three cases to handle the event points. For brevity, let R_i^s, R_i^n, R_i^e, and R_i^w, denote the south, north, east, and west edges of a rectangle R_i. The bisector between two consecutive edges of R_i is a 45° ray emanating from the corner vertex outwards. The bisector of any two edges e_k, e_j is denoted as $B(e_k, e_j)$.

1) West edge of rectangle R_i (Fig. 10). Let the edge be denoted $\overline{a,b}$, where a and b are the northwest and southwest corners of R_i, respectively. We construct bisectors $B(R_i^w, R_i^n)$ and $B(R_i^s, R_i^w)$.
   a) Identify the intersection points a' and b' of $B(R_i^w, R_i^n)$ and $B(R_i^s, R_i^w)$ with the wavefront, respectively. This can be done by binary search in $\mathcal{T}$ for each bisector. The binary search proceeds by discriminating the endpoints of the elements of $\mathcal{T}$ against the bisector. The goal is to identify the pair of consecutive elements in $\mathcal{T}$ whose endpoints lie on opposite sides of the bisector. Then a' and b' can be easily identified as the intersection points of the respective bisector and the segment joining the corresponding pairs of endpoints. In Fig. 10, the elements of $\mathcal{T}$ are depicted thickened. Let e_a and e_b denote the edges owning a' and b'.
   b) Update the Voronoi diagram constructed so far by explicitly constructing the portion of the wavefront between a' and b'. Recall that this corresponds to the line segments joining the endpoints of the elements of $\mathcal{T}$ between a' and b'.
   c) Update $\mathcal{T}$ by removing those spike bisectors between a' and b'. The spike events associated with these spike bisectors can be either *deleted* from the priority queue now or *ignored* later when they get extracted from the priority queue.
   d) Two new spike bisectors $B(e_a, R_i^n)$ and $B(R_i^s, e_b)$ are created with startpoints a' and b', respectively. Moreover, at most two new spike events defined by the intersection of bisector $B(R_i^s, e_b)$ and its lower neighbor and by the intersection of bisector $B(e_a, R_i^n)$ and its upper neighbor will be created and inserted into the priority queue. Note that the priority values of these spike events should be recorded correctly.
   e) Update $\mathcal{T}$ by inserting R_i^n, R_i^s, and the new spike bisectors $B(e_a, R_i^n)$ and $B(R_i^s, e_b)$. In Fig. 10, these two new spike bisectors, $B(e_a, R_i^n)$ and $B(R_i^s, e_b)$, are shown with an arrowhead.
2) East edge of rectangle R_i (Fig. 11). In this case, the two elements R_i^s and R_i^n must be present in $\mathcal{T}$ in consecutive order. These two elements in $\mathcal{T}$ need to be deleted from $\mathcal{T}$ and replaced by two new bisector elements $B(R_i^e, R_i^s)$ and $B(R_i^n, R_i^e)$, respectively (shown with arrowheads in Fig. 11). As in case 1(d), these two newly created bisectors should be checked against their neighboring elements to see if they would create spike events to be inserted in the priority queue with the appropriate priority values.
3) Spike event s (Fig. 12). s is the intersection of two spike bisectors, say $B(R_k^u, R_j^v)$ and $B(R_j^v, R_i^z)$ for some wavefront rectangles R_i, R_j, and R_k, where u, v, z stand for north, east, or south. If either of the two bisectors is not currently in $\mathcal{T}$ [due to case 1(c)], then we discard s as it is no longer relevant. Otherwise, the two bisectors

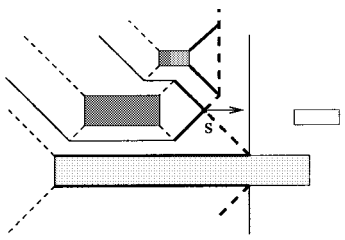


Fig. 12. A spike event.

$B(R_k^u, R_j^v)$ and $B(R_j^v, R_i^z)$ must be adjacent in $\mathcal{T}$. Both bisectors should be deleted from $\mathcal{T}$ and replaced by a new spike bisector $B(R_k^u, R_i^z)$. (The Voronoi cell of their common owner R_j^v becomes part of the final diagram.) This new spike bisector should be checked against its neighboring elements for possible intersections, creating spike events to be inserted in the priority queue with the appropriate priority values.

The data structure to store the Voronoi diagram can be any standard data structure for planar subdivisions. See, for example, the *doubly connected edge list* [15] or the *quad edge* data structure [7]. However, for the critical area computation, there is no need to explicitly maintain the Voronoi diagram. Here, for a rectangle R, it is enough to keep $\mathrm{reg}(R)$ as a linked list of Voronoi edges (bisectors) bounding $\mathrm{reg}(R)$. The algorithm can be summarized with the pseudocode given below. The notation follows the previous discussion. Given a priority queue Q, operation $\mathrm{MIN}(Q)$ returns the element in Q of minimum priority value and deletes it from Q. Operation *intersection*(b_1, b_2) determines the intersection point (if any) between two bisectors b_1 and b_2. Operation *intersection*(*wavefront*, b) determines the intersection point between the wavefront and a bisector b by performing binary search in $\mathcal{T}$ as explained in Step 1(a) above. For a spike bisector b in $\mathcal{T}$, let *prev*(b) and *next*(b) denote the elements in $\mathcal{T}$ preceding and following b, respectively

Procedure L_∞-VORONOI
begin

1. create an edge event for every vertical edge of a rectangle and place them into the event-list Q;
2. $\mathcal{T} = \emptyset$
3. **while** ($Q \neq \emptyset$) **do**
   begin
4. $q = MIN(Q)$;
5. **if** (q is a west edge event) **then**
6. $R =$ rectangle of which q is the west edge;
7. $a' =$ intersection(*wavefront*, $B(R^w, R^n)$);
8. $b' =$ intersection(*wavefront*, $B(R^w, R^s)$);
9. $W = \{w \mid w =$ Voronoi edge corresponding to a segment of the wavefront between a' and $b'\}$;
10. Initialize $reg(R)$ to W;
11. Update $reg(R')$ for every rectangle R' inducing an edge in W;
12. Delete the elements of $\mathcal{T}$ between a' and b';
13. $b_1 = B(e_a, R^n)$; (*$e_a =$ the owner of a'*);
14. $b_2 = B(R^s, e_b)$, (*$e_b =$ the owner of b'*);
15. Insert b_1, R^n, R^s, b_2 in $\mathcal{T}$;
16. $s_1 =$ intersection$(b_1, prev(b_1))$;
17. $s_2 =$ intersection$(b_2, next(b_2))$;
18. Insert in Q spike events corresponding to s_1, s_2;
19. **else if** (q is an east edge event) **then**
20. $R =$ rectangle of which q is the east edge;
21. Delete R^n and R^s from $\mathcal{T}$;
22. Insert $b_1 = B(R^n, R^e), b_2 = B(R^e, R^s)$ in $\mathcal{T}$;
23. $s_1 =$ intersection$(b_1, prev(b_1))$;
24. $s_2 =$ intersection$(b_2, next(b_2))$;
25. Insert in Q spike events corresponding to s_1, s_2;
26. **else if** (q is a valid spike event) **then**
27. $b_1 =$ the upper bisector inducing q;
28. $b_2 =$ the lower bisector inducing q;
29. R_i, R_j, R_k : rectangles whose edges induce q; (*$b_1 == B(R_i, R_j)$; $b_2 == B(R_j, R_k)$;*)
30. **if** $(R_i \neq R_j)$ **then**
31. update $reg(R_i)$ and $reg(R_j)$ with the Voronoi edge corresponding to b_1;
32. **if** $(R_j \neq R_k)$ **then**
33. update $reg(R_j)$ and $reg(R_k)$ with the Voronoi edge corresponding to b_2;
34. delete b_1, b_2 from $\mathcal{T}$;
35. $b =$ the bisector emanating from q defined by by the distinct owners of b_1 and b_2, R_i and R_k;
36. insert b in $\mathcal{T}$;
37. $s_1 =$ intersection$(b, prev(b))$;
38. $s_2 =$ intersection$(b, next(b))$;
39. Insert in Q spike events corresponding to s_1, s_2;
40. **else**
    continue;

end
end

Lemma 3: The Voronoi diagram in the L_∞-metric for a set of rectangles can be computed in $O(n \log n)$ time.

Proof: The number of event points—including spike events—is $O(n)$, and the number of elements maintained in the sweepline status is also $O(n)$ (see [3]). At each event point, standard operations associated with priority queues and height-balanced trees are performed (e.g., *insert*, *delete*, and *delete-min*) which are known to take $O(\log m)$ time, where m is the size of the queue or the tree. In case of a west edge event, points a' and b' can be determined in $O(\log m)$ time by binary search in $\mathcal{T}$ (where m is the size of $\mathcal{T}$), as explained in the previous discussion [Step 1(a)]. Thus, the overall time complexity is $O(n \log n)$.

As in [3], the algorithm updates the wavefront at each event point. As it was discussed above, the structure of the wavefront remains the same between any two consecutive events. Thus, the correctness of the algorithm follows. □

Once the Voronoi diagram is available, we can compute the second-order subdivision within each Voronoi cell in a similar fashion. Fig. 13 shows the second-order partitioning in the Voronoi cell of the middle rectangle R_0. The Voronoi cell of R_0 is shown shaded. The partitioning line segments are depicted in thicker lines, and they are just extensions of those Voronoi edges that are incident on a Voronoi vertex on the boundary of $\mathrm{reg}(R_0)$. The owners of the second-order cells—drawn in thicker lines—are rectangle edges that are

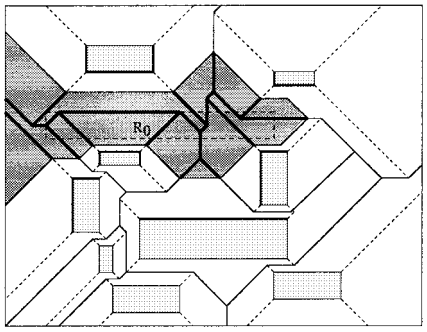


Fig. 13. Second-order Voronoi diagram in the Voronoi cell of R.

associated with the Voronoi edges on the boundary of $\mathrm{reg}(R_0)$. The computation of the second-order subdivision is done independently in every Voronoi cell. Thus, we concentrate on the computation of a single cell, say $\mathrm{reg}(R_0)$.

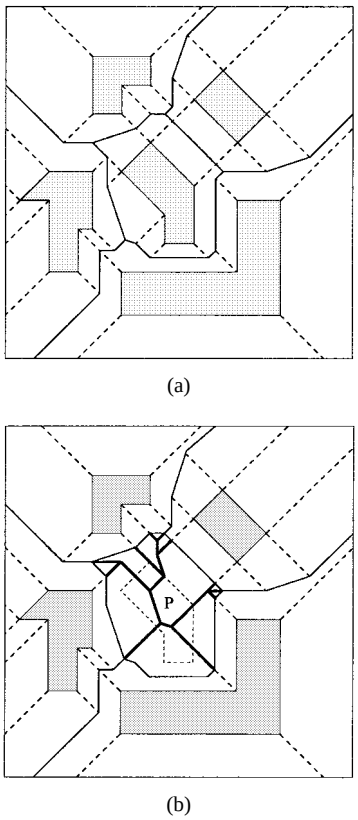
We first identify the neighboring cells and the edges that, together with an edge of R_0, define a Voronoi edge on the boundary of $\mathrm{reg}(R_0)$. These are the only edges involved in the computation of the second-order partition within the Voronoi cell of R_0. The identification can be easily done from the Voronoi edges bounding $\mathrm{reg}(R_0)$. We simply compute the Voronoi diagram of these edges and truncate it within the interior of $\mathrm{reg}(R_0)$. As before, we will use a vertical sweepline sweeping from left to right. The abscissae of the endpoints of site edges will be sorted in ascending order and placed in the event-list. The sweepline status contains at most *two* edges of a rectangle, one south edge and one north edge (since we are partitioning only one cell), plus a number of spike bisectors. As before, the adjacent elements in the sweepline status induce *spike events* which correspond to potential Voronoi vertices. The handling of the event points is exactly the same. Once the Voronoi cell of one edge is computed, it gets truncated by the boundary of $\mathrm{reg}(R_0)$. A Voronoi edge is always truncated by the incident Voronoi vertex along the boundary of $\mathrm{reg}(R_0)$. The time complexity and the correctness of this algorithm follow from the previous discussion.

For the critical area computation, there is no need to explicitly maintain the whole Voronoi diagram. As the plane sweep proceeds, as soon as the Voronoi cell of an individual polygon is computed we can directly compute the second-order subdivision within the cell and derive the portion of critical area due to that cell. Then, the Voronoi cell can be discarded. Thus, at any instant, we only need to have available the Voronoi cells of polygons currently contributing segments to the wavefront. Such a Voronoi cell can be maintained as a linked list of the bounding Voronoi edges.

VII. Critical Area Computation in Ortho-45 Layouts

In this section, we extend the result of Section V to *ortho-45* layouts, i.e., layouts containing shapes with edges of slope ± 1 in addition to orthogonal edges.

As it was shown in [13], the L_∞ Voronoi diagram of ortho-45 polygons contains edges in eight orientations: vertical, horizontal, (± 1)-slope, ($\pm 1/3$)-slope, and (± 3)-slope lines. Fig. 14(a) illustrates the L_∞ Voronoi diagram of ortho-45



(a)

(b)

Fig. 14. The Voronoi diagram of ortho-45 polygons.

shapes. Unlike the rectilinear case, now there are vertices that have Voronoi cells of their own. These are vertices incident to two 45° edges and vertices incident to acute angles. Within the Voronoi cell of a vertex v, distance is measured according to the vertical or horizontal distance from v depending on which quadrant of point v contains the cell. Fig. 14(b) illustrates the second-order subdivision within the cell of polygon P.

The decomposition into rectangles and triangles of the Voronoi cell of a vertical or horizontal edge (or vertex) is obtained similarly to Section V. Moreover, Voronoi edges are classified into red and blue in the same manner. The only difference now is that the hypotenuse of some orthogonal triangles may have slope ± 3 or $\pm 1/3$ in addition to ± 1. Thus, the formulas of Lemma 1 for nonisosceles triangles get slightly modified as follows.

Lemma 4: The critical area within a nonisosceles diverging triangle T_{div}, and a nonisosceles converging triangle T_{cnv} in the decomposition of the (second-order) Voronoi cell of an orthogonal edge, is given by the following formulas, using the "r_0^2/r^3" defect density distribution:

$$A_c(T_{\mathrm{div}}) = r_0^2/2(3\ln(r_k/r_j) - l/r_k) \quad (5)$$

$$A_c(T_{\mathrm{cnv}}) = r_0^2/2\big(l/r_j - 3\ln(r_k/r_j)\big) \quad (6)$$

where l is the size of the edge of T_{div} and T_{cnv} parallel to e, r_k, r_j, $r_k > r_j$, are the maximum and the minimum critical radius of their vertices.

Proof: The difference from the derivation of Lemma 1 is that $l = 3w$, where l, w denote the length of the vertical and the horizontal edge of T_{cnv} and T_{div}, respectively. (Note that the

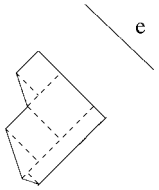


Fig. 15. The decomposition of the Voronoi cell of a 45° edge.

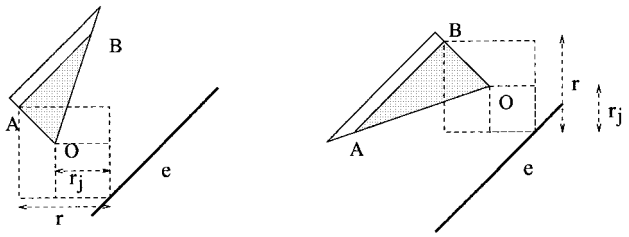


Fig. 16. A diverging triangle $T_{\rm div}$ in the cell of a 45° edge.

slope of the hypotenuse is ± 3 for a vertical owner and $\pm 1/3$ for a horizontal owner.) For a defect radius r, $r_j < r \leq r_k$, $l_r = 3w_r$, i.e., $l_r = 3(r - r_j)$. The formulas follow in the same fashion as Lemma 1. □

For a (second-order) Voronoi cell V of a (-1)-slope edge e, the decomposition of V into trapezoids is obtained by drawing lines of slope $+1$ from the vertices of V. The decomposition of each trapezoid into triangles and rectangles is obtained by drawing (-1)-slope lines from its vertices. Triangles are characterized as diverging or converging similarly to Section V. Fig. 15 shows the decomposition for the cell of a (-1)-slope edge in the Voronoi diagram of Fig. 14. Similarly, for the Voronoi cell of a $(+1)$-slope edge. The formulas of Lemma 1 for Voronoi cells of 45° edges are modified as follows.

Lemma 5: The critical area within a rectangle $\mathcal{R}$, a diverging triangle $T_{\rm div}$, and a converging triangle $T_{\rm cnv}$ in the decomposition of the (second-order) Voronoi cell of a 45° edge, is given by the following formulas, using the "r_0^2/r^3" defect density distribution

$$A_c(\mathcal{R}) = r_0^2/\sqrt{2}(l/r_j - l/r_k) \tag{7}$$

$$A_c(T_{\rm div}) = r_0^2/\sqrt{2}\Big(t\sqrt{2}\ln(r_k/r_j) - l/r_k\Big) \tag{8}$$

$$A_c(T_{\rm cnv}) = r_0^2/\sqrt{2}\Big(l/r_j - t\sqrt{2}\ln(r_k/r_j)\Big) \tag{9}$$

where l is the size of the edge of $\mathcal{R}$, $T_{\rm div}$ and $T_{\rm cnv}$ parallel to e, r_k, r_j, $r_k > r_j$, are the maximum and the minimum critical radius of their vertices, and t is a factor reflecting the slope of the hypotenuse; $t = 2$, if the hypotenuse has slope ± 3, $\pm 1/3$, and $t = 1$, otherwise.

Proof: Let e be the 45° edge, which is the owner of the (second-order) Voronoi cell V. The area of the critical region within $\mathcal{R}$ for a defect radius r, $r_j \leq r \leq r_k$, is $A(r, \mathcal{R}) = lw_r$, where l is the length of the side parallel to e, and w_r is the length of the portion of the side perpendicular to e within L_∞ distance r from e. Here, $w_r = \sqrt{2}(r - r_j)$, thus, $A(r, \mathcal{R}) = \sqrt{2}l(r - r_j)$. The formula follows in the same manner as in Lemma 1.

Let $T_{\rm div}$ be a diverging triangle and let l, w be the length of the side parallel and perpendicular to e, respectively. Let us assume without loss of generality that e has slope $+1$. Then the hypotenuse of $T_{\rm div}$ can have slope $s = 0, 1/3, 3, \infty$. Fig. 16 illustrates diverging triangles with hypotenuse of slope $s = 3, 1/3$. Given a defect radius r, $r_j < r \leq r_k$, the area of critical region within T is the area of triangle OAB (see Fig. 16), i.e., $A(r, T_{\rm div}) = 1/2 l_r w_r$, where $w_r = |OA|$ and $l_r = |AB|$. Clearly, $w_r = \sqrt{2}(r - r_j)$. For $s = 0, \infty$, $l_r = w_r$. To determine l_r for $s = 3, 1/3$, let us treat O as the origin of the coordinate system and let us rotate the coordinate system by $\phi = -45°$ so that AB becomes horizontal. In the new coordinate system the slope of the hypotenuse OB is $s' = (s-1)/(s+1) = \pm 1/2$. Thus, $l_r = |AB| = 2w_r$. Thus, $A(r, T_{\rm div}) = t(r - r_j)^2$, where $t = 2$, if $s = 3, 1/3$ and $t = 1$, otherwise. Following the derivation of Lemma 1, we get $A_c(T_{\rm div}) = r_0^2 t(\ln(r_k/r_j) + (r_j - r_k)/r_k)$. But $l = t\sqrt{2}(r_k - r_j)$. Thus, $A_c(T_{\rm div}) = r_0^2/\sqrt{2}(t\sqrt{2}\ln(r_k/r_j) - l/r_k)$.

For a converging triangle $T_{\rm cnv}$, $A_c(T_{\rm cnv}) = A_c(\mathcal{R}) - A_c(T_{\rm div})$ as in Lemma 1. □

The Voronoi edges of cell V are colored *red* and *blue* in a manner similar to the rectilinear case. However, there may be 45° edges that do not contribute to the formulas of Lemma 5 which are not colored. In particular, 45° Voronoi edges parallel to the owner e are colored red if the interior and the owner of the cell lie on opposite sides of the edge; otherwise they are colored blue. 45° Voronoi edges perpendicular to the owner are unclassified with respect to this owner and they do not contribute to critical area within V. Unclassified edges are colored *neutral*. The remaining Voronoi edges of V induce the hypotenuse of a triangle and they are colored red or blue depending on whether the triangle is diverging or converging, respectively. Any boundary edges bounding V must be colored blue.

As in the rectilinear case, the total critical area is derived by adding the critical areas within every (second-order) Voronoi cell. Every Voronoi edge bounds exactly two cells and receives the same coloring (with the exception of neutral) with respect to both cells. Note that an edge may be colored red or blue with respect to one cell and neutral with respect to the other. In the following we generalize Theorem 1. The term *prime* is used to denote Voronoi edges that are parallel to their owner, i.e., Voronoi edges corresponding to bisectors of parallel edges.

Theorem 2: Given the second-order L_∞ Voronoi diagram of polygons in an ortho-45 layer of a circuit layout C, and assuming that defects are squares following the "r_0^2/r^3" defect density distribution, the critical area for shorts in that layer is given by the formula shown at the bottom of the next page, where l_i and r_i denote the length and the critical radius of a prime Voronoi edge r_k, r_j, $r_k > r_j$, denote the maximum and the minimum critical radius of a nonprime Voronoi (or boundary) edge, and l_b, r_b denote the length and the critical radius of a boundary edge. Factor $h_i = 1$ for orthogonal prime Voronoi edges, and $h_i = \sqrt{2}$ for 45° prime Voronoi edges. Factor $f_i = 1, 2, 1/2, 7/2$ depending on the slope of the Voronoi edge and its owners. In particular, $f_i = 1$ for a 45° Voronoi edge with orthogonal owners, $f_i = 1/2$ for a 45° Voronoi edge with a 45° owner, $f_i = 2$ for an orthogonal Voronoi edge of 45° owners, and $f_i = 7/2$ for a (± 3, $\pm 1/3$)-slope Voronoi edge. The first two summations are taken

over all red and all blue prime Voronoi edges, respectively. The third and forth summations are taken over all red and all blue nonprime Voronoi edges, respectively. The last two summations are taken over all blue boundary edges, the former over prime boundary edges and the latter over nonprime ones.

Proof: As in Theorem 1, by summing up critical areas within every (second-order) Voronoi cell, terms corresponding to internal decomposition vertices or edges cancel out. By Lemma 5, (7), prime 45° Voronoi edges contribute $(\sqrt{2}l_i)/(2r_i)$ to critical area for each owner, while orthogonal prime Voronoi edges contribute $l_i/(2r_i)$. Thus, the values of h_i can be derived.

The values of factor f_i are derived as follows. For a 45° nonprime Voronoi edge with orthogonal owners, f_i is clearly one (as in the rectilinear case). A 45° nonprime Voronoi edge induced by a 45° owner and an orthogonal edge contributes zero to critical area for the 45° owner (it is colored neutral), and $1/2\ln(r_k/r_j)$ for the orthogonal owner. Thus, $f_i = 1/2$. An orthogonal Voronoi edge of 45° owners contributes $\ln(r_k/r_j)$ for each owner [Lemma 5, (8) and (9)], thus, $f_i = 2$ in this case. A (± 3, $\pm 1/3$)-slope Voronoi edge contributes $(3/2)\ln(r_k/r_j)$ because of its orthogonal owner (Lemma 4) and $2\ln(r_k/r_j)$ because of the 45° owner (Lemma 5). Thus, $f_i = 7/2$. The terms for boundary edges are derived similarly. □

VIII. A More Efficient Grid Method

In [18], Wagner and Koren consider the area of the layout as a set of I grid points of resolution γ. The input polygons (shapes) are also treated as collection of grid points. They compute the critical radius at every grid point by visiting on average $\sqrt{I}$ other grid points, i.e, total time complexity is $O(I^{1.5})$. Once the critical radius at every grid point is known the integral of the total critical area can be easily computed; the error depends on the grid resolution γ.

Imagine now that the shape grid points are sources of waves that start propagating at the same time with the same speed. The wave has the id of the net, where the source grid point belongs. Waves of the same id are considered identical (although they come from different sources). The propagation is done from one grid point t to its immediate neighbors. Assuming that defects are squares, all eight grid points neighboring t, vertically, horizontally, and diagonally, are considered to be the immediate neighbors of t. During the propagation exactly two distinct waves are allowed to overlap at any grid point.

The wave propagation will assign to every grid point t a pair of distance labels (d_1, d_2), where d_1 indicates the distance from the source grid point whose wave is the first one to reach t and d_2 indicates the distance from the source grid point of different id whose wave is the second one to reach t. Clearly d_2 is the critical radius at point t. During the propagation we allow exactly two distinct waves to overlap. When a grid point p is visited by wave w the following happens: 1) if p has already been visited by wave w or if p has already been visited by two other distinct waves the propagation stops at p. 2) Otherwise, p updates its labels (updates label d_1, if w is the first wave to reach p or label d_2, if w is the second wave to reach p) and propagates w to its immediate neighbors (grid points) increasing the distance label by one.

This construction corresponds to an implicit mechanical construction of the second-order L_∞ Voronoi diagram and thus correctness follows from the previous discussion. Every grid point is visited by the waves exactly twice and thus the critical radius at every grid point can be computed in $O(I)$ time. Then the critical area can be computed as described in [18].

IX. Conclusion

We have shown how to compute and use the second-order L_∞ Voronoi diagram of rectilinear polygons to efficiently compute critical area for shorts in a single layer. The use of the L_∞ metric is equivalent to assuming defects of size r to be squares of side $2r$. The value of critical area can then be used in evaluating yield. Furthermore, we can use the Voronoi diagram to identify the places in the layout that are most vulnerable to spot defects. Note that Voronoi edges provide immediate information about the partial critical area caused by the corresponding pair of layout edges. In other words, we can readily obtain information about those layout edges that contribute most to critical area. By slightly perturbing such edges, we may be able to reduce the value of critical area and, thus, increase yield. Note that after moving an edge the Voronoi diagram can be updated dynamically in time proportional to the number of changes caused to the diagram because of the move. The changes correspond to the insertion of new Voronoi edges and the deletion of those Voronoi edges affected by the move. Thus, the Voronoi diagram approach is suitable for an interactive critical area tool. Note also that the *topographic map* of [18] essentially provides a bitmapped version of the more critical Voronoi edges. The Voronoi diagram approach can also be extended to layouts with shapes in arbitrary orientations as it is shown in [13].

Acknowledgment

The authors would like to thank Dr. D. Ostapko for reading through an earlier draft and providing useful comments.

$$A_c = r_0^2 \cdot \left(\sum_{\substack{\text{red,}\\ \text{prime}}} h_i(l_i/r_i) - \sum_{\substack{\text{blue,}\\ \text{prime}}} h_i(l_i/r_i) + \sum_{\substack{\text{red,non-}\\ \text{prime}}} f_j \ln(r_k/r_j) - \sum_{\substack{\text{blue,non-}\\ \text{prime}}} f_j \ln(r_k/r_j) - 1/2 \sum_{\substack{\text{blue,}\\ \text{prime}}} l_b/r_b - \sum_{\substack{\text{blue,non-}\\ \text{prime}}} \ln(r_k/r_j) \right)$$

REFERENCES

[1] F. Aurenhammer, "Voronoi diagrams: A survey of a fundamental geometric data structure," *ACM Comput. Survey*, vol. 23, pp. 345–405, 1991.

[2] F. Aurenhammer and R. Klein, "Voronoi diagrams," in *Textbook on Computational Geometry*, J. Sack and G. Urrutia, Eds. ch. 18, to be published.

[3] F. Dehne and R. Klein, "The big sweep: On the power of the wavefront approach to Voronoi diagrams," *Algorithmica*, vol. 17, pp. 19–32, 1997.

[4] A. V. Ferris-Prabhu, "Modeling the critical area in yield forecast," *IEEE J. Solid-State Circuits*, vol. SC-20, pp. 874–878, Aug. 1985.

[5] ———, "Defect size variations and their effect on the critical area of VLSI devices," *IEEE J. Solid-State Circuits*, vol. SC-20, pp. 878–880, Aug. 1985.

[6] S. Gandemer, B. C. Tremintin, and J. J. Charlot, "Critical area and critical levels calculation in IC yield modeling," *IEEE J. Solid-State Circuits*, vol. 35, no. 2, pp. 158–166, Feb. 1988.

[7] L. Guibas and J. Stolfi, "Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams," *ACM Trans. Graphics*, vol. 4, no. 2, pp. 74–123, Apr. 1985.

[8] I. Koren, "The effect of scaling on the yield of VLSI circuits," in *Yield Modeling and Defect Tolerance in VLSI Circuits*, W. R. Moore, W. Maly, and A. Strojwas, Eds.. Bristol U.K.: Adam-Hilger, 1988, pp. 91–99.

[9] D. T. Lee, "On k-nearest neighbor Voronoi diagrams in the plane," *IEEE Trans. Comput.*, vol. C-31, pp. 478–487, June 1982.

[10] W. Maly, "Modeling of lithography related yield losses for CAD of VLSI circuits," *IEEE Trans. Computer-Aided Design*, vol. CAD-4, pp. 166–177, July 1985.

[11] ———, "Computer aided design for VLSI circuit manufacturability," *Proc. IEEE*, Feb. 1990, pp. 356–392.

[12] W. Maly and J. Deszczka, "Yield estimation model for VLSI artwork evaluation," *Electron Lett.*, vol. 19, no. 6, pp. 226–227, Mar. 1983.

[13] E. Papadopoulou and D. T. Lee, "L_∞ Voronoi diagrams and applications to VLSI layout and manufacturing," Manuscript, Extended Abstract in *Proc. 9th Int. Symp. Algorithms and Computation, Lecture Notes in Computer Science*, 1998, vol. 1533, pp. 9–18.

[14] J. P. de Gyvez and C. Di, "IC defect sensitivity for footprint-type spot defects," *IEEE Trans. Computer-Aided Design*, vol. 11, pp. 638–658, May 1992.

[15] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*. New York: Springer-Verlag, 1985.

[16] C. H. Stapper and R. J. Rosner, "Integrated circuit yield management and yield analysis: Development and implementation," *IEEE Trans. Semiconduct. Manufact.* vol. 8, pp. 95–101, Feb. 1995.

[17] C. H. Stapper, "Modeling of defects in integrated circuits photolithographic patterns," *IBM J. Res. Develop.*, vol. 28, no. 4, pp. 461–475, 1984.

[18] I. A. Wagner and I. Koren, "An interactive VLSI CAD tool for yield estimation," *IEEE Trans. Semiconduct. Manufact.*, vol. 8, pp. 130–138, Feb. 1995.

[19] H. Walker, *VLASIC System User Manual, Release 1.3*, Carnegie Mellon University (CMU), Pittsburgh, PA, Aug. 1990.

[20] H. Walker and S. W. Director, "VLASIC: A yield simulator for integrated circuits," *IEEE Trans. Computer-Aided Design*, vol. CAD-5, pp. 541–556, Oct. 1986.

Evanthia Papadopoulou received the B.S. degree in mathematics form the University of Athens, Athens, Greece, and the M.S. degree in computer science from the University of Illinois at Chicago. She received the Ph.D. degree in computer science from Northwestern University, Evanston, IL, in December 1995.

She spent the summer of 1993 as a Visiting Researcher at the Institute of Information Sciences, Academia Sinica, Taiwan. In 1996, she joined the IBM T. J. Watson Research Center, Yorktown Heights, NY, where she is currently a Research Staff Member in the Department of Design Automation and Verification. Her research interests include design and analysis of algorithms, computational geometry, and VLSI computer-aided design.

D. T. Lee (S'76–M'78–SM'84–F'92) received the B.S. degree in electrical engineering from the National Taiwan University, Taipei, Taiwan, in 1971, and the M.S. and Ph.D. degrees in computer science from the University of Illinois at Urbana-Champaign in 1976 and 1978, respectively.

Since 1978, he has worked at Northwestern University, Evanston, IL, where he is currently a Professor of the Department of Electrical and Computer Engineering. He spent one year (August 1989–August 1990) working as Program Director for Computer and Computation Theory Program, Division of Computer and Computation Research, National Science Foundation, Washington, D.C. His research interests include design and analysis of algorithms, computational geometry, VLSI layout, parallel and distributed computing, web-based computing, algorithm visualization, and compliant controller for active suspension and vibration control. He is Editor of *Algorithmica, Networks, Computational Geometry: Theory & Applications*, *ACM Journal of Experimental Algorithmics*, Managing Editor of the *International Journal of Foundations of Computer Science*, Managing Editor of the *International Journal of Computational Geometry & Applications Series*, and Editor of *Lecture Notes Series on Computing* (Singapore: World Scientific) since 1990.

Dr. Lee is a Member of SIAM and Fellow of ACM.