Higher Order Voronoi Diagrams of Segments for VLSI Critical Area Extraction

Evanthia Papadopoulou

IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA
Athens University of Economics and Business, Athens 10434, Greece
evanthia@acm.org

Abstract. We address the problem of computing critical area for opens in a circuit layout in the presence of multilayer loops and redundant interconnects. The extraction of critical area is a fundamental problem in VLSI yield prediction. We first model the problem as a graph problem and we solve it efficiently by exploiting its geometric nature. We introduce the *opens Voronoi diagram of polygonal objects*, a generalization of Voronoi diagrams based on concepts of higher order Voronoi diagrams of segments. Higher order Voronoi diagrams of segments had not received much attention in the literature. The approach expands the Voronoi critical area computation paradigm [1,2,3] with the ability to accurately compute critical area for missing material defects even in the presence of loops and redundant interconnects spanning over multiple layers.

1 Introduction

Catastrophic yield loss of integrated circuits is caused to a large extent by random particle defects interfering with the manufacturing process resulting in functional failures such as open or short circuits. All yield models for random manufacturing defects focus on critical area, a measure reflecting the sensitivity of the design to random defects during manufacturing (e.g. [4,5,6]). Reliable critical area extraction is essential for today's IC manufacturing especially when DFM, i.e., design for manufacturability, initiatives are under consideration.

The critical area of a circuit layout on a layer A 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 a circuit failure and $D(r)$ is the density function of the defect size. $D(r)$ has been estimated as $D(r) = r_0^2/r^3$, where r_0 is some minimum optically resolvable size. Critical area analysis is typically performed on a per layer basis and results are combined to estimate total yield.

In this paper we focus on critical area extraction for *open faults* resulting from broken interconnects generalizing upon the results of [2]. Opens are net-aware, that is, a defect is considered a fault only if it actually *breaks* a net. A net is

T. Tokuyama (Ed.): ISAAC 2007, LNCS 4835, pp. 716–727, 2007.

said to be *broken* if there exist terminal points that get disconnected. In order to increase design reliability and reduce the potential for open circuits designers are increasingly introducing redundant interconnects creating interconnect loops that may span over a number of layers (see e.g. [7]). Redundant interconnects reduce the potential for open faults at the expense of increasing the potential for shorts. The ability to perform trade-offs is important requiring accurate critical area computation for both.

In this paper we first model the problem as a graph problem and we solve it efficiently by exploiting the geometric nature of it. We formulate the *opens Voronoi diagram*, a generalization of Voronoi diagrams based on concepts of higher order Voronoi diagrams of segments. To the best of our knowledge higher order Voronoi diagrams of segments have not received much attention in the literature with the recent exception of [8] for the farthest segment Voronoi diagram. Once the opens Voronoi diagram is available the entire critical area integral for opens can be computed analytically in linear time similarly to [1,9,2].

For computational simplicity we have adapted the L_∞ metric throughout this paper [9]. This is consistent with the common practice of modeling defects as squares to facilitate critical area computation. For simplicity figures are depicted in Manhattan geometry, however, the method is general and it is applicable to layouts of arbitrary geometry as well as other metrics of potential interest such as the Euclidean or the k-gon metric. The algorithms presented in this paper have been integrated into the IBM Voronoi Critical Area Analysis tool (*Voronoi CAA*) which is used extensively by IBM manufacturing for the prediction of yield. For results on the industrial use of our tool and comparisons with previously available tools see [10].

2 Review of L_∞ Voronoi Diagrams for Opens

The Voronoi diagram of a set of polygonal sites in the plane is a partitioning of the plane into regions, one for each site, called *Voronoi regions*, such that the Voronoi region of a site s is the locus of points closer to s than to any other site. The Voronoi region of s is denoted as $reg(s)$ and s is called the *owner* of $reg(s)$. The boundary that borders two Voronoi regions is called a *Voronoi edge*, and consists of portions of *bisectors* between the owners of the neighboring cells. The bisector of two polygonal objects (such as points, segments, polygons) is the locus of points equidistant from the two objects. The point where three or more Voronoi edges meet is called a *Voronoi vertex*. The combinatorial complexity of the Voronoi diagram is linear in the number and complexity of the sites. In the interior of a simple polygon the Voronoi diagram is also called *medial axis*[1]. Any point p on the boundary of $reg(s)$ is weighted by $w(p) = d(p, s)$. The disk D centered at p of radius $w(p)$ is *empty*, that is, no site intersects the interior of D.

The L_∞ distance is used throughout this paper. The L_∞ distance between two points $p = (x_p, y_p)$ and $q = (x_q, y_q)$ is $d(p, q) = \max\{|x_p - x_q|, |y_p - y_q|\}$. In

[1] There is a minor difference in the definition which we ignore in this paper (see [11]).

the presence of additive weights, the (weighted) distance is $d_w(p,q) = d(p,q) + w(p) + w(q)$, where $w(p)$ and $w(q)$ denote the weights of points p, q respectively. In case of a weighted line l, the (weighted) distance between a point t and l is $d_w(t,l) = \min\{d(t,q)+w(q), \forall q \in l\}$. The (weighted) bisector between polygonal elements s_i, s_j is $b(s_i,s_j) = \{y \mid d_w(s_i,y) = d_w(s_j,y)\}$. In this paper we use the term *core element*, i.e., *core segment* and *core point*, to denote a portion of interest along a bisector (such as a medial axis segment, a Voronoi edge or a Voronoi vertex). Fig. 1 illustrates examples of core segments. The endpoints and the open line segment portion of a core segment are always differentiated and they are treated as distinct entities.

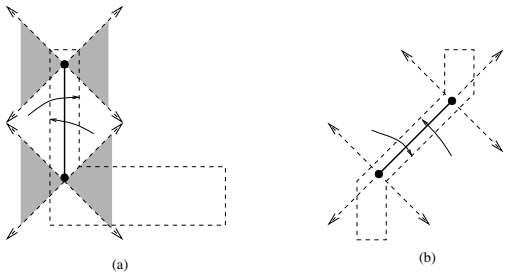

Fig. 1. The regions of influence of the core elements of a core segment

In L_∞ core segments can be treated as additively weighted ordinary segments. Let s be a core segment induced by the polygonal elements e_l, e_r, that is, s is portion of bisector $b(e_l, e_r)$. Every point p along s is weighted with $w(p) = d(p,e_l) = d(p,e_r)$. The 45° rays[2] emanating from the endpoints of s partition the plane into the regions of influence of either the open core segment portion or the core endpoints. (In the Euclidean metric the corresponding rays would be rays perpendicular to e_l and e_r). Fig.1 illustrates the partitioning of space induced by a core segment in the L_∞ metric. Shaded regions are equidistant from both the core endpoint and the open core segment. In this paper equidistant regions are always assigned to the core endpoint. In the region of influence of a core point p distance is measured in the ordinary weighted sense, that is, for any point t, $d_w(t,p) = d(t,p) + w(p)$. In the region of influence of an open core segment s distance in essence is measured according to the farthest polygonal element defining s, that is, $d_w(t,s) = d(t,e_l)$ where e_l is the polygonal element at the opposite side of the line through s than t (indicated by arrows in Fig.1). In L_∞ this is equivalent to the ordinary weighted distance from s. The bisector between two core elements, and therefore the Voronoi diagram of a set of core elements, can now be defined as usual. The (weighted) Voronoi diagram of *core* medial axis segments was introduced in [2] as a solution to the critical area computation problem for a simpler notion of an open based solely on geometric information called *break*.

[2] A 45° ray is a ray of slope ±1.

3 A Graph Representation for Nets

From a layout perspective a net N is a collection of interconnected shapes spanning over a number of layers. The portion of N on a given layer X is denoted as $N_X = N \cap X$ and consists of a number of connected components interconnected through different layers. Every connected component is a collection of overlapping polygons that can be unioned into a single shape (a simple one or one with holes). Some of the shapes constituting net N are designated as *terminal shapes* representing the entities that the net must interconnect. A net remains *functional* as long as all terminal shapes comprising the net remain interconnected. Otherwise the net is said to be *broken*. Fig. 2(a) illustrates a simple net N spanning over two metal layers, say M1 and M2, where M2 is illustrated shaded. The two contacts illustrated as black squares have been designated as terminal shapes. In Fig. 2(b) defects that create opens are illustrated as dark squares and defects that cause no fault are illustrated hollow in dashed lines. Note that hollow defects do break wires of layer M1 but they do not create an open as they do not break net N.

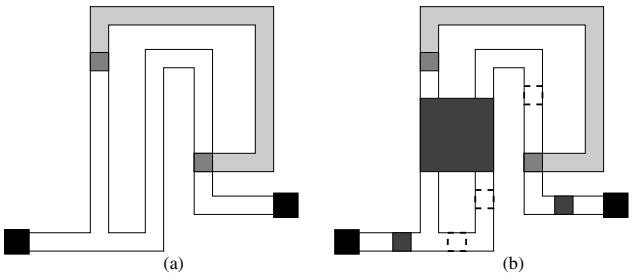

Fig. 2. (a) A net N spanning over two layers. (b) Dark defects create opens while transparent defects cause no faults.

We define a compact graph representation for N, denoted $G(N)$, as follows. There is a graph node for every connected component of N on a conducting layer. A node containing terminal shapes is designated as a terminal node. Two graph nodes are connected by an edge iff there exists at least one contact or via connecting the respective components of N. To build $G(N)$ some net extraction capability needs to be available. Net extraction is a well studied topic beyond the scope of this paper. For the purposes of this paper we assume that $G(N)$ can be available for any net.

To perform critical area computation on a layer A we derive the *extended graph* of N on layer A, denoted as $G(N, A)$, that can be obtained from $G(N)$ by expanding all components of N_A by their medial axis. For every via or contact introduce an approximate point along the medial axis representing that via or contact, referred to as a *via-point*, and a graph edge connecting the via-point with the node of the connecting component of N. If a contact or via has been

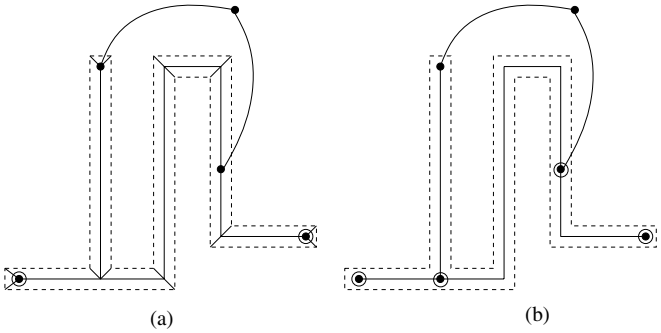

Fig. 3. The net graph of Fig. 2 before (a) and after (b) cleanup of trivial parts

designated as terminal shape, designate also the corresponding via point as terminal. In the presence of via clusters we can keep only one via point representing the entire cluster. Any portion of the medial axis induced by edges of terminal shapes is also identified as terminal. Fig. 3a illustrates $G(N, A)$, where $A = M1$, for the net of Fig. 2. Terminal points are indicated by hollow circles. Dashed lines represent the original M1 polygon and are not part of $G(N, M1)$.

Given $G(N, A)$ we can detect *biconnected components, bridges* and *articulation points* using *depth-first search* (DFS) as described in [12,13]. For our problem we only maintain some additional terminal information to determine whether the removal of a vertex or edge actually *breaks* $G(N, A)$, i.e., whether it disconnects $G(N, A)$ leaving terminals in both sides. For this purpose we chose the root of the DFS tree to be a terminal node or terminal point and at every node i of the DFS tree we keep a flag indicating whether the subtree rooted at i contains a terminal point. Any bridges or any articulation points whose removal does not disconnect terminals of $G(N, A)$ are called *trivial*. Similarly any biconnected component incident to only trivial articulation points that contains no terminal points is called trivial. Trivial bridges, trivial articulation points and trivial biconnected components can be easily determined and removed from the graph with no effect on the net connectivity regarding opens. In the following we assume that $G(N, A)$ has been cleaned up from all trivial parts, and thus, the removal of any bridge or any articulation point results in a fault. Fig. 3(b) illustrates the net graph after the cleaning of all trivial parts. Hollow circles indicate terminal and articulation points. The graph has exactly one bi-connected component.

4 Modeling Net-Aware Opens

In this section we formalize the definition of a net-aware open.

Definition 1. *A* minimal open *is a defect D that breaks a net N and D has minimal size, that is, if D is shrunk by $\epsilon > 0$ then D no longer breaks N. D breaks N if any two terminal shapes of N get disconnected or if a terminal shape*

itself gets destroyed. A minimal open is called strictly minimal *if it contains no other minimal open in its interior. An* open *is any defect entirely covering a minimal open. The size of a minimal open centered at a point t is called the* critical radius *of t.*

Definition 2. *The center point of an open D is called a* generator point *for D and it is weighted with the size (radius) of D. A segment formed as a union of generator points is called a* generator segment *or simply a generator.*

Definition 3. *The* core *of the extended net graph $G(N, A)$ on layer A, denoted $core(N, A)$, is the set of all medial axis vertices, including articulation, via, and terminal points, and all medial axis edges, except the standard-45° edges*[3]*. $G(N, A)$ is assumed to have been cleaned up from any trivial components, trivial bridges, or trivial articulation points. The union of $core(N, A)$ for all nets N is denoted as $core(A)$.*

A core segment is assumed to consist of three distinct core elements: two endpoints and an open line segment. Given a net N, $core(N, A)$ induces a unique decomposition of the portion of N on layer A into well defined wire segments. In particular, any core element s induces a wire segment $R(s) = \cup_{p \in s} R(p)$, where $R(p)$ denotes the disk (the square in L_∞) centered at core point p having radius $w(p)$. Those wire segments may overlap and their union reconstructs N_A (excluding the trivial portions of N_A). In Fig. 3b all depicted medial axis vertices and segments constitute $core(N, A)$. The dark shaded disks of Fig. 2 are strictly minimal opens.

Definition 4. *A defect D is classified as order $k, k \geq 1$, if D overlaps k wire segments as induced by k distinct core elements of $core(A)$. The center point of D is classified as a kth order generator, $k \geq 1$. In case of core elements equidistant (in weighted sense) from the center of D, a kth order defect is allowed to overlap more than k wire segments.*

Definition 5. *A* cut *is a collection of core elements $C \subset core(N, A)$, such that $G(N, A) - C$ is disconnected leaving non-trivial articulation or terminal points in at least two different sides. A cut C of k elements is called minimal if $C - \{c\}$ is not a cut for any element $c \in C$*

Lemma 1. *The set of 1st order generators for strictly minimal opens on layer A consists exactly of the all the bridges, terminal edges, articulation points, and terminal points of $G(N, A) \cap core(N, A)$ for every net N.*

5 A Higher Order Voronoi Digram Modeling Opens

The Voronoi diagram for opens on layer A, referred to as the *opens Voronoi diagram*, is a subdivision of the layer into regions such that each region reveals

[3] The term standard-45° refers to portions of bisectors of axis parallel lines of slope ± 1.

the *critical radius* for opens for every point in that region. Recall that the critical radius of a point t, $r_c(t)$, is the size of the smallest defect centered at t causing a circuit failure. A circuit failure here corresponds to an open. For any point t in a region $reg(h)$ of the opens Voronoi diagram the critical radius of t should be derived as a distance function from the owner h, specifically $r_c(t) = d_w(t, h)$ for $h \in core(A)$. In the following we formulate the opens Voronoi diagram as a higher order Voronoi diagram of core segments.

Consider the (weighted) Voronoi diagram of all core elements on layer A, denoted as $V(A)$. If there are no loops associated with layer A then all elements of $core(A)$ must be 1st order generators and $V(A)$ must provide the opens Voronoi diagram on layer A (see also [2]). Fig. 4 illustrates $V(A)$ for the net graph of Fig. 2. To model opens appropriately we follow some additional conventions for $V(A)$ as follows: A region equidistant from a core segment and its endpoint is always assigned to the endpoint. All regions of 1st order generators are colored red. Coloring a region red indicates that the critical radius of every point in the region is determined by the owner of that region.

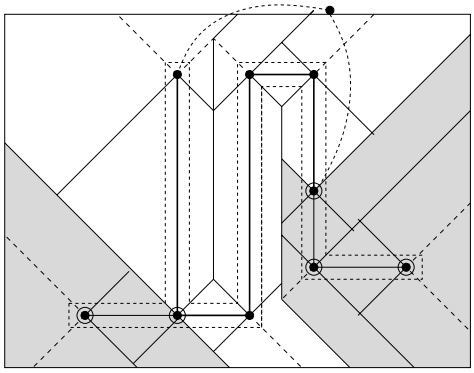

Fig. 4. The Voronoi diagram $V(A)$ for a net N on layer A

Let us now define the order-k Voronoi diagram of layer A, denoted as $V^k(A)$. For $k = 1$, $V^k(A) = V(A)$. A non-red region of $V^k(A)$ is a locus of points with the same k nearest neighbors (in a weighted sense) among the core elements in $core(A)$. A red region of $V^k(A)$ is a locus of points with the same r, $1 \leq r \leq k$, nearest neighbors among the core elements in $core(A)$, such that the set C of those r core elements constitutes a minimal cut for some net N. If $|C| > 1$ the red Voronoi region $reg(C)$ is further subdivided into finer subregions by the farthest Voronoi diagram of C, denoted $V_f(C)$. Fig. 5 illustrates $V^2(A)$ for the net of our example. Voronoi regions of $V^2(A)$ are illustrated in solid lines and red regions are illustrated shaded. The thick dashed lines indicate $V_f(C)$ for cuts $C, |C| = 2$.

There are two types of red regions in $V^k(A)$: those that are expanded red regions of $V^{k-1}(A)$, referred to as *old red regions*, and *new red regions* of cuts

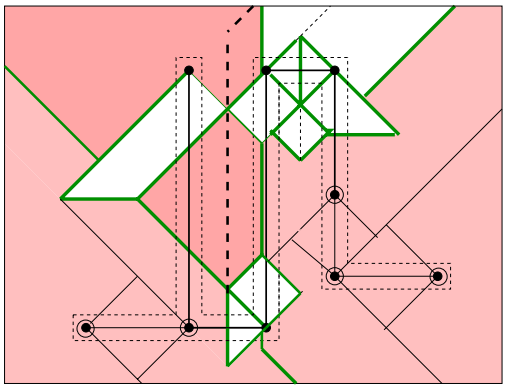

Fig. 5. The 2nd order Voronoi diagram $V^2(A)$

determined in $V^k(A)$. Clearly any red region of $V^k(A)$ remains red in $V^{k+1}(A)$. In Fig. 5 new red regions are illustrated darker. The thick dashed axis-parallel segments in the new red regions of Fig. 5 are the 2nd order generators for minimal opens.

Theorem 1. *The Voronoi diagram for opens on layer A is the minimum order-k Voronoi diagram of core(A), $V^k(A)$, such that all regions of $V^k(A)$ are colored red. Any region $reg(H)$ such that H consists of more than one core element is further subdivided into finer regions by $V_f(H)$, the farthest Voronoi diagram of H. The critical radius for any point t in a Voronoi region $reg(H)$ is $r_c(t) = \max\{d_w(t,h), h \in H\}$. In particular, if t is in the subregion $reg(h) \subset reg(H) \cap V_f(H)$ then $r_c(t) = d_w(t,h)$.*

Corollary 1. *The higher order generators for strictly minimal opens on layer A are exactly the farthest Voronoi edges and vertices (except the standard-45° Voronoi edges) of the opens Voronoi diagram on layer A constituting the farthest Voronoi subdivisions in the Voronoi region of any cut C of size $|C| > 1$.*

To the best of our knowledge higher order Voronoi diagrams of segments have not received much attention in the literature unlike higher order Voronoi diagrams of points, see e.g. [11,14,15]. The problem can have different flavors depending on whether segments are treated as closed entities or whether the open portions of segments are treated as distinct from their endpoints. In this paper we only deal with the latter interpretation as this is the one modeling our application.

5.1 Computing the Opens Voronoi Diagram

To obtain the Voronoi diagram for opens we can adapt the simple iterative process to obtain higher order Voronoi diagrams of points (see e.g. [11]) as we are primarily interested in small values of k. The main difference with the standard

case of points is that an open core segment s does not exist in the region of its endpoint p, that is, s can not be considered as a higher order nearest neighbor in $reg(p)$. Furthermore, in L_∞, there can be regions equidistant from more than one element. As a result, unlike the Euclidean metric, the k-tuples owning two neighboring Voronoi regions may differ in more than one element.

We first obtain $V(A)$ by plane sweep modifying the algorithm of [9] to accommodate weights and the convention of assigning priority to endpoints as opposed to the open portion of segments. Note that weights are special ensuring no disconnected Voronoi regions. The latter convention can be accommodated either by modifying the original algorithm to include the bisectors (45° rays in L_∞) between the open portion of core segments and their endpoints, or after the original Voronoi diagram is constructed by drawing the additional bisectors in linear time. Intersections among the additional bisectors can be resolved arbitrarily. All Voronoi subregions associated with the same core point p are unified into a single Voronoi region for p. The properties and the asymptotic combinatorial complexity of the Voronoi diagram remain the same.

Let's now assume that $V^k(A), k \geq 1$, is available. We show how to compute $V^{k+1}(A)$. Let $reg(H)$ be a non-red region of $V^k(A)$ and let $s(H)$ denote the superset of H defined as H union all open core segments incident to the core points in H. Let $N(H)$ denote the union of all core elements owning Voronoi regions neighboring the regions of elements in $s(H)$. Compute the (weighted) L_∞ Voronoi diagram of $N(H) - s(H)$ and truncate it within the interior of $reg(H)$; this gives the $(k+1)$-order subdivision within $reg(H)$. Each $(k+1)$-order subregion within $reg(H)$ is attributed to a $(k+1)$-tuple $H \cup \{c\}$, where $c \in N(H) - s(H)$. Once the $(k+1)$-order subdivision has been performed within the non-red regions of $V^k(A)$ we remove the edges and vertices of $V^k(A)$ that are not part of $V^{k+1}(A)$, merge the incident $(k+1)$-order subregions of $V^k(A)$ into the $(k+1)$-order Voronoi regions of $V^{k+1}(A)$, and determine the red regions of $V^{k+1}(A)$. Unlike the standard higher order Voronoi diagram case, not all Voronoi edges of $V^k(A)$ are necessarily deleted from $V^{k+1}(A)$. In the following we give the details of this process.

Let $reg(H)$ be a non-red region of $V^k(A)$ and let c, $c \notin H$, be a core element inducing a $(k+1)$-order subregion in $reg(H)$. Let $reg(H \cup \{c\})$ denote the union of all $(k+1)$-order subregions of $V^k(A)$ owned by $H \cup \{c\}$. Recall that no $(k+1)$-order subdivision is performed within red regions of $V^k(A)$, that is, no portion of $reg(H \cup \{c\})$ is red other than possibly some bounding Voronoi edges. Any Voronoi element of $V^k(A)$ in the interior of $reg(H \cup \{c\})$ gets deleted from $V^{k+1}(A)$ (unless colored red, see below) and all subregions of $reg(H \cup \{c\})$ get merged into a new $(k+1)$-order region of $V^{k+1}(A)$. It remains to determine whether $H \cup \{c\}$ forms a cut for the biconnected component B such that $c \in B$. There are two cases:

1. If $reg(H \cup \{c\})$ is incident to an already red Voronoi region $reg(R)$ of $V^k(A)$ such that $R \subset H \cup \{c\}$ then clearly $H \cup \{c\}$ forms a cut. Then $reg(H \cup \{c\})$ is colored red and $reg(H \cup \{c\})$ gets merged within $reg(R)$. Since no portion of

$reg(H \cup \{c\})$ is already red in $V^k(A)$ it is not hard to see that the portion of $V^k(A)$ in the interior of $reg(H \cup \{c\})$ is in fact the corresponding portion of the farthest Voronoi diagram of R, $V_f(R)$. We say that that the region of the cut R *expands* into $reg(H \cup \{c\})$ keeping R as the sole owner of the expanded region. The portion of $V^k(A)$ in the interior of $reg(H \cup \{c\})$ remains as a finer subdivision of the expanded region.

2. Otherwise we need to check whether $H \cup \{c\}$ forms a new cut (see below), i.e., whether $reg(H \cup \{c\})$ becomes a new red region of $V^{k+1}(A)$. If $H \cup \{c\}$ is indeed a new cut of biconnected component B, let $C = B \cap (H \cup \{c\})$; C is assigned as the owner of $reg(H \cup \{c\})$, which is now denoted simply as $reg(C)$, and it is colored red. The interior of $reg(C)$ gets partitioned into finer subregions by $V_f(C)$, the farthest Voronoi diagram of C, as given by the portion of $V^k(A)$ in the interior of $reg(C)$. It is not hard to see that no information is lost by assigning C as the owner of $reg(H \cup \{c\})$ as no core element in $(H \cup \{c\}) - C$ can be the farthest one among elements of $H \cup \{c\}$ for any point $t \in reg(H \cup \{c\})$. In fact any element of H that is not represented in the (complete) farthest Voronoi diagram of C, can be excluded from C.

The following Lemma can be derived using the properties of standard higher order Voronoi diagrams of points (see [11]).

Lemma 2. *The opens Voronoi diagram on layer A has size $O(k(n-k))$, where n denotes the number of polygonal edges on layer A, and k is the maximum number of iterations performed in the construction of the diagram until all regions are colored red. The number k depends on the connectivity of $G(N, A)$.*

A simple (almost brute force) algorithm to determine new cuts of $V^{k+1}(A)$ is as follows. Let $reg(H)$ be a non-red region of $V^k(A)$ and let B be a biconnected component associated with set H. That is, $B \cap H \neq \emptyset$ and there is a Voronoi edge bounding $reg(H)$ induced by core elements b, h such that $b \in B - H$ and $h \in H$. Remove the elements of H from B and determine new non-trivial bridges, articulation points and biconnected components of $B - H$. Clearly $H \cup \{c\}$ is a new cut of B if and only if c is a new non-trivial bridge or articulation point of $B - H$. It is now easy to determine any new cut associated with the non-red Voronoi edges or vertices bordering $reg(H)$ in $V^k(A)$. Note that any new cut of $V^{k+1}(A)$ corresponds to at least one Voronoi element of $V^k(A)$.

To determine the new cuts of $V^2(A)$ (i.e., generators of order 2) a much faster algorithm could be obtained by partitioning biconnected components of $G(N, A)$ into *triconnected components* (see [16]). However this is not easily generalizable to $k > 2$. Many biconnected components in actual VLSI designs are expected to have low connectivity to the extent of being simple cycles. For simple cycles the problem is easy and can be answered using a simple coloring scheme on the DFS tree of the corresponding biconnected component.

The time complexity of the entire algorithm is described in the following lemma.

Lemma 3. *The Voronoi diagram for opens on a VLSI layer A can be computed in time $O(k^2 n \log n)$ plus a total $O(k^2 n^2)$ time to determine cuts associated with higher order generators, where k is the maximum number of iterations performed and n is the complexity of layer A. In case of biconnected components forming simple cycles or if the maximum number of iterations is bounded by $k = 2$ the bound simplifies to $O(n \log n)$.*

6 A Hausdorff Voronoi Diagram for Opens

Once the set of cuts $\mathcal{C}$ claiming a region in the opens Voronoi diagram on layer A have been identified, the opens Voronoi diagram can be interpreted as the Hausdorff Voronoi diagram of $\mathcal{C}$. Given a cut C and a point t, the Hausdorff distance between t and C simplifies to the maximum (weighted) distance of t from any core element in C, i.e., $d_h(t, C) = \max\{d_w(t, c), c \in C\}$. The Hausdorff Voronoi diagram of a set of cuts S is the Voronoi diagram of S under the Hausdorff distance, where the Hausdorff Voronoi region of a cut C is $reg(C) = \{t \mid d_h(t, C) \leq d_h(x, C_j), C_j \in S\}$ (for the definition of an ordinary Hausdorff Voronoi diagram see e.g [17,3]). Assuming that some superset of cuts $S \supseteq \mathcal{C}$ can be identified, the Hausdorff Voronoi diagram of S provides an alternative definition for the opens Voronoi diagram on a layer A. This observation can help reduce the number of iterations in computing the final opens Voronoi diagram and speed up the construction in practice. Namely, once a sufficient set of cuts $\mathcal{C}'$ has been identified the iteration can stop and the Hausdorff Voronoi diagram of $\mathcal{C}'$ can be directly computed in the non-red portion of the current $V^k(A)$. Furthermore, one can localize the higher order Voronoi diagram computation by applying the iterative process to identify cuts to each biconnected component independently. Computing the Hausdorff Voronoi diagram of cuts for all biconnected components union the 1st order generators provides the opens Voronoi diagram. Practical limits on the number of iterations for each biconnected component can be easily imposed to gain speed with only minimal potential loss in accuracy (if any). For a plane sweep algorithm to compute the Hausdorff Voronoi diagram for clusters of points see [3].

Often it is desirable to compute one critical area value combining interconnect opens on layer A and *via-blocks* on the neighboring via layers into a single estimate of critical area for *missing material defects*. A via-block is a defect entirely destroying a connection (a via or cluster of vias) between neighboring conducting layers. The problem of computing critical area for via-blocks reduces to computing a Hausdorff Voronoi digram of polygons representing clusters of redundant vias (see [2,3]). To compute the combined Voronoi diagram for missing material defects we simply need to substitute $V(A)$, with $V_h(A')$, the Hausdorff Voronoi diagram of all core elements on layer A union clusters of vias on the neighboring via layers. Voronoi regions of via-clusters represent via-blocks and are always colored red. $V_h(A')$ can be computed by plane sweep by adapting the plane sweep construction of [3]. The iterative process to compute new cuts and the final Voronoi diagram for missing material defects remains similar.

Acknowledgments

The author would like to thank Dr. L. F. Heng of IBM T.J. Watson Research center for helpful discussions on the definition of a net graph. The IBM Voronoi EDA group including R. Allen, S.C. Braasch, J.D. Hibbeler and M.Y. Tan are greatly acknowledged for jointly developing, supporting, and expanding the Voronoi Critical Area Analysis Tool.

References

1. Papadopoulou, E., Lee, D.T.: Critical area computation via Voronoi diagrams. IEEE Transactions on Computer-Aided Design 18(4), 463–474 (1999)
2. Papadopoulou, E.: Critical area computation for missing material defects in VLSI circuits. IEEE Transactions on Computer-Aided Design 20(5), 583–597 (2001)
3. Papadopoulou, E.: The Hausdorff Voronoi diagram of point clusters in the plane. Algorithmica 40, 63–82 (2004)
4. Stapper, C.H., Rosner, R.J.: Integrated circuit yield management and yield analysis: Development and implementation. IEEE Trans. Semiconductor Manufacturing 8(2), 95–102 (1995)
5. Stapper, C.H.: Modeling of defects in integrated circuit photolithographic patterns. IBM J. Res. Develop. 28(4), 461–475 (1984)
6. Ferris-Prabhu, A.: Defect size variations and their effect on the critical area of VLSI devices. IEEE J. of Solid State Circuits 20(4), 878–880 (1985)
7. Kahng, A.B., Liu, B., Mandoiu, I.I.: Non-tree routing for reliability and yield improvement. IEEE Trans. on Comp. Aided Design of Integrated Circuits and Systems 23(1), 148–156 (2004)
8. Aurenhammer, F., Drysdale, R., Krasser, H.: Farthest line segment Voronoi diagrams. Information Processing Letters (100), 220–225 (2006)
9. Papadopoulou, E., Lee, D.: The l_∞ Voronoi diagram of segments and VLSI applications. International Journal of Computational Geometry and Applications 11(5), 503–528 (2001)
10. Maynard, D.N., Hibbeler, J.D.: Measurement and reduction of critical area using Voronoi diagrams. In: Advanced Semiconductor Manufacturing IEEE Conference and Workshop (2005)
11. Lee, D.T.: On k-nearest neighbor Voronoi diagrams in the plane. IEEE Trans. Comput. C-31(6), 478–487 (1982)
12. Tarjan, R.: Depth-first search and linear graph algorithms. SIAM Journal on Computing 1, 146–160 (1972)
13. Hopcroft, J., Tarjan, R.: Efficient algorithms for graph manipulation. Comm. ACM 16(6), 372–378 (1973)
14. Aurenhammer, F.: A new duality result concerning Voronoi diagrams. Discrete and Cpomputational Geometry 5, 243–254 (1990)
15. Chazelle, B., Edelsbrunner, H.: An improved algorithm for constructing kth order Voronoi diagrams. IEEE Transactions on Computers C-36, 1349–1354 (1987)
16. Hopcroft, J., Tarjan, R.: Dividing a graph into triconnected components. SIAM Journal on Computing 2, 135–158 (1973)
17. Aurenhammer, F., Klein, R.: Voronoi diagrams. In: Sack, J., Urrutia, G. (eds.) Handbook of Computational Geometry, pp. 201–290. Elsevier, Amsterdam (2000)