USI Technical Report Series in Informatics

Net-aware Critical Area Extraction for Opens in VLSI Circuits via Higher Order Voronoi Diagrams

Evanthia Papadopoulou[1]

[1] Faculty of Informatics, Università della Svizzera italiana, Switzerland

Abstract

We address the problem of computing critical area for open faults (opens) in a circuit layout in the presence of multilayer loops and redundant interconnects. The extraction of critical area is the main computational bottleneck in predicting the yield loss of a VLSI design due to random manufacturing defects. We first model the problem as a geometric graph problem and we solve it efficiently by exploiting its geometric nature. To model open faults we formulate a new geometric version of the classic min-cut problem in graphs, termed the *geometric min-cut problem*. Then the critical area extraction problem gets reduced to the construction of a generalized Voronoi diagram for open faults, based on concepts of higher order Voronoi diagrams. The approach expands the Voronoi critical area computation paradigm [5, 16–19, 22, 28] 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. The generalized Voronoi diagrams used in the solution are combinatorial structures of independent interest.

Report Info

Published
June 2010

Number
USI-INF-TR-2010-6

Institution
Faculty of Informatics
Università della Svizzera italiana
Lugano, Switzerland

Online Access
www.inf.usi.ch/techreports

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. Yield loss due to random manufacturing defects has been studied extensively in both industry and academia and several yield models for random defects have been proposed (see e.g., [8, 25, 26]). The focus of all models is the concept of *critical area*, a measure reflecting the sensitivity of a design to random defects during manufacturing. Reliable critical area extraction is essential for today's IC manufacturing especially when DFM (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. The defect density function has been estimated as follows [8, 12, 25, 29]:

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

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. Using typical values for p, q, and c, the widely used defect size distribution is derived,

$D(r) = r_0^2/r^3$. (r_0 is typically smaller than the minimum feature size thus, $D(r)$ is ignored for $r < r_0$). 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* (*opens*) resulting from broken interconnects. Open faults are *net-aware*, that is, a defect causes a fault if and only if it actually *breaks* a net leaving *terminals* disconnected. A net is said to be broken if at least one of its terminals gets disconnected. In order to increase design reliability and reduce the potential for open circuits designers have been introducing redundant interconnects creating interconnect loops that may span over a number of layers (see e.g. [11]). Redundant interconnects reduce the potential for open faults at the expense of increasing the potential for shorts. Therefore, the ability to perform trade-offs is important requiring accurate critical area computation for both opens and shorts. A critical area extraction tool that fails to take loops into consideration would falsely penalize designs with redundant interconnects by (erroneously) overestimating the actual critical area for opens while (correctly) registering the increase in critical area for shorts.

In previous work on critical area extraction for open faults interconnects have been typically assumed acyclic, that is, a defect *breaking* any conducting path is considered a fault (see e.g. [23], [7,16]). This assumption was adequate at the time, however, it is no longer realistic. An exception is [24] where loops were being detected and treated as immune to open faults. Critical area in [24], however, was considered strictly over each layout shape ignoring all critical regions expanding in the free space or over other shapes resulting in underestimation of critical area that can be arbitrarily large.

Existing methods for critical area extraction focus mostly on shorts while opens have been typically treated as a dual problem. The methods can be roughly grouped into the following categories:

1. Monte Carlo simulation, the oldest most widely used technique for critical area extraction [30].

2. Iterative shape-shifting techniques that compute $A(r)$ for several different values of r independently and then use these values to extract the total critical area integral, see e.g., [2, 7, 23, 24, 32]. Shape shifting techniques are typically based on shape manipulation tools providing operations such as *expand-shape-by-r* and *find-area* for a given defect radius r (with the exception of [7,24] that are based on plane sweep and work strictly for Manhattan geometries). For opens, the reverse process *shrink-shape-by-r* is typically used, which however fails to capture several aspects of open faults. Layout sampling in combination with shape shifting techniques were introduced in [1].

3. The Voronoi method [5, 16, 18, 20, 22, 28] which is using analytical formulas to extract the entire critical area after deriving a subdivision of the layout into regions that reveal the critical radius (size of smallest defect) of every point. The critical area integral is typically computed with no error in a single pass of the layout using $O(n \log n)$ type of scan line algorithms. In addition the Voronoi method can be combined effectively with layout sampling techniques such as in [1, 4], for a fast critical area estimate at the chip level.

4. A grid based method introduced in [29] (time complexity improved in [22]).

In this paper we focus on the Voronoi method and we expand it with the ability to detect loops and report true open faults that are net-aware. Loops are not assumed to be immune to open faults as they can still be broken by defects and thus they can contribute to critical area. To model open faults we first model a VLSI net as a graph of geometric nature and we introduce a geometric version of the classic *min-cut* problem in graphs, termed the *geometric min-cut* problem. We then solve the problem efficiently by exploiting its geometric nature. We formulate a generalized Voronoi diagram for open faults, termed the *opens Voronoi diagram*, which is based on concepts of *higher order* and *Hausdorff* Voronoi diagrams (see [18]). The Voronoi diagram for open faults is a combinatorial structure interesting on its own right. Once the opens Voronoi diagram on a given layer is available the entire critical area integral can be computed analytically, in linear time, using the formulas given in [16, 20, 22].

The algorithms presented in this paper have been integrated in the IBM Voronoi Critical Area Analysis tool (*Voronoi CAA*) [5, 28] currently used in production mode throughout IBM manufacturing. For results on the early industrial use of Voronoi CAA and comparisons with previously available tools see [14]. An important difference between the Voronoi method and previous geometric approaches to critical area extraction is that it can directly compute the entire critical area integral for all possible defect radii without any repetition. Other methods typically compute $A(r)$ for a specific defect radius r and then repeat for a number of radii until they extract the entire critical area integral (see e.g. [2, 7, 23, 24, 32]). In contrast the Voronoi method computes the critical area integral directly, using analytical formulas, resulting in no integration error and in a fast deterministic method. If in addition the value of $A(r)$, for some specific defect radius r, is desirable, it can be easily extracted from the corresponding Voronoi diagram requiring no additional effort. For a fast critical area estimation at the chip level the Voronoi method can be combined easily with sampling techniques, either random [1] or deterministic [4], that sample a number of windows over the layout applying the Voronoi critical area extraction method to a fraction only of the entire design, deriving a reliable estimate of critical area.

The methods presented in this paper are applicable to layouts of arbitrary geometry, and do not assume a Manhattan layout. A Manhattan layout however would result in a simpler implementation. For simplicity, figures are depicted in Manhattan geometry. Our implementation of Voronoi CAA assumes ortho-45[1] geometries in the layout. Throughout this paper defects are modeled as squares, that is, a defect of size r is modeled as a square of radius r i.e., a square of side $2r$. This corresponds to computing critical area in the L_∞ metric[2] (also known as max-norm) instead of the standard Euclidean plane. Square defects are among the most common simplifications found in critical area literature. A formal worst case bound for critical area estimation between the L_∞ and the Euclidean metric i.e., critical area estimation between square and circular defects, is given in [16].

The paper is organized as follows. In Section 2 we review basic concepts of Voronoi diagrams as related to the Voronoi method for critical area extraction that are needed in subsequent sections. In Section 3 we show how to model a net as a graph of geometric nature to facilitate the modeling of net-aware opens and the extraction of critical area. In Section 4 we give formal definitions for a net-aware open and the opens Voronoi diagram and define the geometric min-cut problem. In Section 5 we model the opens Voronoi diagram as a special higher order Voronoi diagram of segments. In Section 6 we discuss the algorithm to compute the opens Voronoi and give practical simplifications. Finally in Section7 we provide experimental results.

Once Voronoi regions of the opens Voronoi diagram are available, the critical area integral is extracted using the formulas given in [16, 20, 22]. Since this is a known technique presented in previous literature we refer the reader to [16, 20, 22] and we skip discussion in this paper.

2 Review of concepts of Voronoi Diagrams related to modeling 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 regions. 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 ordinary Voronoi diagram of polygonal sites is linear in the number, more precisely linear in the total combinatorial complexity, of the sites. In the interior of a simple polygon the Voronoi diagram is known as *medial axis*[3] of the polygon.

Throughout this paper we use the L_∞ metric. 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 the presence of additive weights, the (weighted) distance between p and q 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 two polygonal elements s_i and s_j is $b(s_i, s_j) = \{y \mid d_w(s_i, y) = d_w(s_j, y)\}$. Using the L_∞ metric for critical area analysis corresponds to modeling defects as squares.

In L_∞, Voronoi edges and vertices can be treated as additively weighted line segments. For brevity and in order to differentiate with ordinary line segments we use the term *core segment* or *core element* to denote any portion of interest along an L_∞ Voronoi edge or vertex. We also use the term *standard*-45° edges to refer to Voronoi edges of slope ± 1 that correspond to bisectors of axis parallel lines. Fig. 1 illustrates examples of core segments. The endpoints and the open line segment portion of a core segment are differentiated and they are treated as distinct entities.

Let s be a core segment induced by the polygonal elements e_l, e_r, that is, s is portion of bisector $b(e_l, e_r)$.

[1] A layout is called ortho-45 if all geometrics are axis parallel or have slope ± 1.

[2] The L_∞ 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., $d(p,q) = \max\{|x_p - x_q|, |y_p - y_q|\}$.

[3] There is a minor difference in the definition which we ignore in this paper (see [13]).

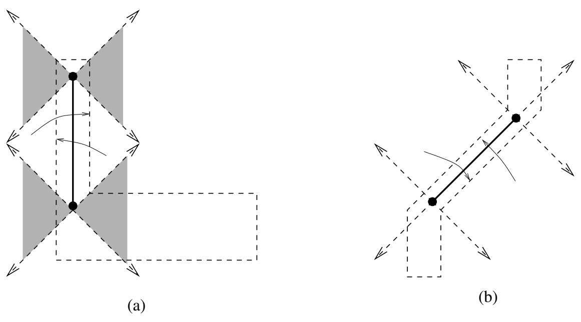


Figure 1: The regions of influence of the core elements of a core segment.

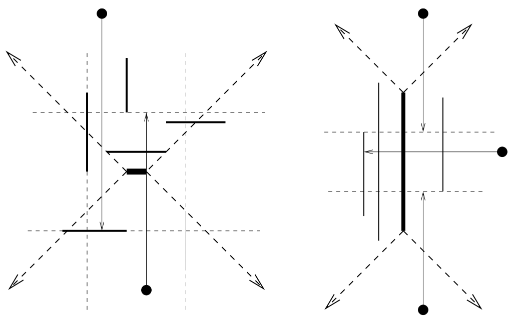


Figure 2: The L_∞ farthest Voronoi diagram of axis parallel segments.

Every point p along s is weighted with $w(p) = d(p, e_l) = d(p, e_r)$. The 45° rays[4] 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. Fig.1 illustrates the partitioning of space induced by a core segment in the L_∞ metric. Shaded regions in Fig.1 are equidistant from both the core endpoint and the open core segment and can be assigned arbitrarily to one of the two. 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. In Fig.1, e_l is indicated by arrows for different points. In L_∞ this is equivalent to the ordinary weighted distance between t and s. The (weighted) bisector between two core elements can now be defined in the ordinary way, always taking the weights of the core elements into consideration. Similarly the (weighted) Voronoi diagram of a set of core elements can be defined as usual, using the definition given above, with the difference that distance between a point t and a core element s is always measured in an additive weighted sense, $d_w(t, s)$. The (weighted) Voronoi diagram of *core* medial axis segments was first introduced in [16] as it provided a solution to the critical area computation problem for a simpler notion of an open (called *break*) that was based solely on geometric information. For Manhattan geometries, core segments are simple (additively weighted) axis parallel line segments and points.

An important variation of Voronoi diagrams is the so called *farthest Voronoi diagram*. The farthest Voronoi diagram of a set of polygonal sites is a partitioning of the plane into regions, such that the farthest Voronoi region of a site s is the locus of points *farther away* from s than from any other site. For typical cases (e.g. points, line segments) the farthest Voronoi diagram is a tree-like structure consisting only of unbounded regions (see e.g. [3, 6, 15]). In the L_∞ metric, when sites are points or axis-parallel segments, the structure of the L_∞ farthest Voronoi diagram is particularly simple, consisting always of exactly four regions. Figure 2 depicts the farthest Voronoi diagram of two sets of axis parallel segments. In both cases the farthest Voronoi diagram consists of an axis parallel segment, shown in bold, (that can degenerate to a point) and four 45°-rays, shown as dashed bold rays, that together partition the plane into four regions (see [16]). In each region, the L_∞ distance to the farthest element is measured as the vertical or horizontal distance to an axis parallel line. In Figure 2 the axis parallel lines indicating farthest distance are depicted as dashed lines. The thin arrows indicate the farthest L_∞ distance of selected points.

[4]A 45° ray is a ray of slope ± 1.

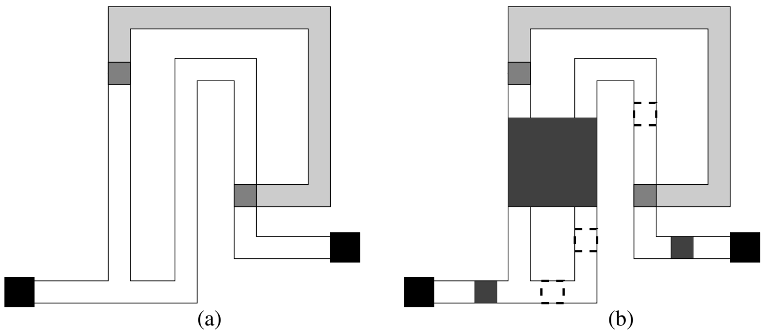


Figure 3: (a) A net N spanning over two layers. (b) Dark defects create opens while transparent defects cause no faults.

3 A graph representation for nets

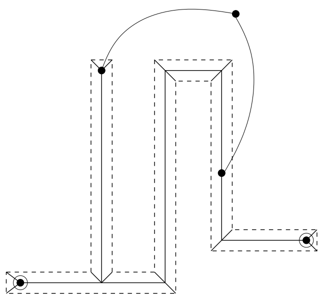
From a layout perspective a net N is a collection of interconnected shapes spanning over a number of layers. The portion of N on a given layer A, $N \cap A$, consists of a number of connected components. Every connected component is a collection of overlapping polygons that can be unioned into a single 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. Terminal shapes typically consist of power buses (collection of shapes representing VDD or GND), gates (intersections of PC and RX shapes), Sources and Drains of Transistors (portions of RX shapes as obtained after subtracting regions overlapping with PC), and pins of macros. Terminal shapes can also be user defined depending on user goals. A net remains *functional* as long as all terminal shapes comprising the net remain interconnected. Otherwise the net is said to be *broken*. Fig. 3(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. 3(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, however, they do not create opens as no terminals get disconnected.

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 if and only if 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. We assume that such capability exists. If not it is not hard to obtain one using a scan line approach that detects intersections among shapes on same and neighboring layers and maintains nets using a *union-find* data structure for efficiency. 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 on layer 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 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. 4a illustrates $G(N,A)$, where $A = M1$, for the net of Fig. 3. Terminal points are indicated by hollow circles. Dashed lines represent the original M1 polygon and they are not part of $G(N,A)$.

Given $G(N,A)$ we can detect *biconnected components*, *bridges* and *articulation points*[5] using *depth-first search* (DFS) as described in [10,27]. 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 at least two different 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*. Any biconnected component incident to only trivial

[5] A biconnected component of a graph G is a maximal set of edges such that any two edges in the set lie on a common simple cycle. An articulation point (resp. bridge) of G is a vertex (resp. edge) whose removal disconnects G.


(a)

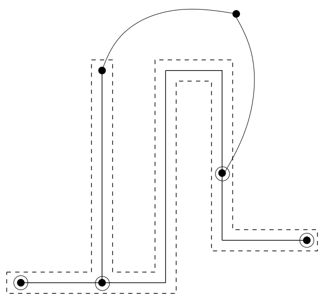

(b)

Figure 4: The net graph of Fig. 3 before (a) and after (b) cleanup of trivial parts.

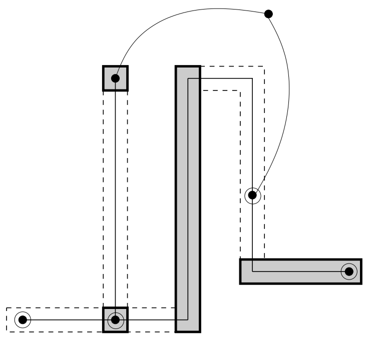


Figure 5: Wire segments as induced by core elements.

articulation points that contains no terminal points is called trivial. Trivial bridges, trivial articulation points and trivial biconnected components can be easily determined during the DFS and they can be 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 always results in a fault. Fig. 4(b) illustrates the net graph of our example after the cleaning of all trivial parts. Hollow circles indicate terminal and articulation points; the graph has exactly one bi-connected component.

Given $G(N,A)$, cleaned from all trivial parts, the collection of medial axis vertices and edges, excluding the standard-45° edges[6], is denoted as $core(N,A)$ and it is referred to as the *core* of net N on layer A ($core(N,A) \subseteq G(N,A)$). In Fig. 4b, all the depicted medial axis vertices and segments constitute $core(N,A)$.

The core of 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 (i.e., a square in L_∞) centered at core point p having radius $w(p)$. Those wire segments may overlap and their union reconstructs all the non-trivial portions of $N \cap A$. Figure 5 illustrates some wire segments as induced by some core segments and core points.

The union of $core(N,A)$ for all nets N on layer A is denoted as $core(A)$. Core elements in $core(A)$ represent all wire segments vulnerable to defects on layer A. Core segments are assumed to consist of three distinct core elements: two endpoints and an open line segment.

4 Modeling net-aware opens

In this section we formalize the intuitive definition of an open that is net-aware and we give definitions for the terminology used throughout this paper.

A defect D *breaks* a net N if D overlaps portions of N such that at least one of its terminal shapes gets disconnected or if a terminal shape itself gets destroyed. Such a defect is called an *open*. More precisely we have the following definitions.

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. An* open *is any defect that entirely overlaps a minimal open. A minimal*

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

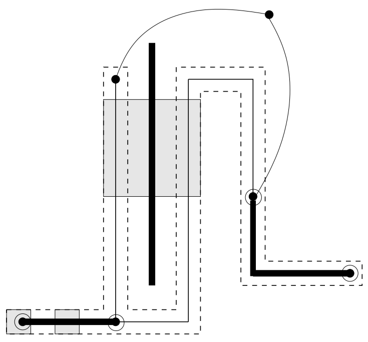


Figure 6: Generators for strictly minimal opens.

open is called strictly minimal *if it contains no other open in its interior.*

In Fig. 3 the dark shaded disks, other than the original via and contact shapes, are strictly minimal opens.

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

Figure 6 illustrates thickened the generators for strictly minimal opens for the net graph of our example. The shaded squares indicate strictly minimal opens.

Recall from Section 3 that the *core* of a net N on layer A, $core(N,A) \subset G(N,A)$, induces a unique decomposition of N into wire segments that are vulnerable to defects. We will slightly abuse terminology and say that a defect D *overlaps* a core element c, $c \in core(A)$, but we will mean that D overlaps the entire width of the wire segment induced by c.

Definition 3 *A* cut *for a net N is a collection C 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. Cut C is called minimal if $C - \{c\}$ is not a cut for any element $c \in C$. A defect that overlaps all elements of cut C is called a* cut-inducing *defect. D is called strictly minimal if D has minimal size, and in addition, it does not entirely overlap any other cut-inducing defect. The centerpoint p of a strictly minimal cut-inducing defect D is called a* generator point *for cut C. If D is also a strictly minimal open then generator point p is called* critical. *The collection of all generator points of cut C is referred to as the* generators(s) *of C.*

The generator of a cut C can consist of *critical* and *non-critical* portions. Critical portions correspond to generators of strictly minimal opens on layer A. Non-critical portions correspond to centers of cut-inducing disks that in addition to overlapping C they also overlap some other cut on layer A, and thus, although they break C, they are not strictly minimal opens.

Definition 4 *Generators of minimal cuts that consist of a single core element are called* first order generators. *Generators of minimal cuts that consist of more than one core element are called* higher order generators. *The set of all critical generators on layer A is denoted as $G(A)$.*

In Figure 6, first order generators are shown as thick core segments; the vertical thick segment in the exterior of polygons is a higher order generator that involves pairs of core elements. By definition we have the following property.

Lemma 1 *All the bridges, terminal edges, articulation points, and terminal points of $G(N,A) \cap core(N,A)$, for any net N, constitute the set of all* first order generators *on layer A, denoted as $G_1(A)$. Generators $G_1(A)$ are all critical.*

By definition, the generator of a minimal cut C that consists of more than one core element must be a subset of the L_∞ farthest Voronoi diagram of C, derived by ignoring the standard-45° edges of the diagram. For Manhattan geometries the generator of any cut C must always be a single axis-parallel segment that can degenerate to a point, see Fig. 2 that depicts the farthest Voronoi diagram of axis parallel segments. Any generator point p of a cut C is weighted with $w(p) = \max\{d_w(p,c), \forall c \in C\}$ i.e., the distance of p from the farthest element in the cut in a weighted sense. The disk D centered at p of radius $w(p)$ is clearly an open. If in addition D is a strictly minimal open then p is a critical generator.

Definition 5 *The* Voronoi diagram for opens *on layer A is a subdivision of layer A into regions such that the* critical radius *of any point t in a region is determined by the owner of the region. The critical radius of a point t, $r_c(t)$, is the size (radius) of the smallest defect centered at t causing an open.*

Figure 11 illustrates the opens Voronoi diagram for the net of Fig. 3. The critical radii of several points are illustrated by arrows.

Theorem 1 *The* Voronoi diagram *for opens on layer A corresponds to the (weighted) Voronoi diagram of the set $G(A)$ of all critical generators for strictly minimal opens on layer A, and it is denoted as $\mathcal{V}(G(A))$.*

Proof. Consider $\mathcal{V}(G(A))$ and let t be a point in the region of a generator g, $g \in G(A)$. By definition, the disk centered at t of radius $d_w(t,g)$, $D(t)$, must entirely overlap a disk centered along a point p on g of radius $w(p)$, $D(p)$. Since p is a generator for strictly minimal opens, $D(p)$ must be a strictly minimal open and therefore $D(t)$ must be an open. Since t is in $reg(g)$, g must be the closest generator to t (in a weighted sense). Thus, if $D(t)$ is shrunk by any positive amount ϵ it will no longer cause an open, as otherwise there would exist some other generator point closer to t than g i.e., t would not be in $reg(g)$. Hence, $D(t)$ is the smallest defect centered at t that causes an open, and thus, $d_w(t,g)$ is the critical radius of t. □

Corollary 1 *Given the opens Voronoi diagram, $\mathcal{V}(G(A))$, the critical radius of any point t in the region of a generator g, $t \in reg(g)$, is $r_c(t) = d_w(t,g)$. If g is a higher order generator of cut C then $r_c(t) = d_w(t,g) =$* $\max\{d_w(t,c), \forall c \in C\}$.

The Voronoi diagram for opens provides a solution to the following problem, termed the *geometric min cut* problem: Given is a collection of geometric graphs that have portions embedded on a plane A, such as the collection of the expanded net graphs $G(N,A)$. The embedded portions on plane A are vulnerable to defects that may form *cuts* on the given graphs. The size of a geometric cut C at a given point t is given by the size of the smallest defect centered at t that overlaps all elements in C (not the number of edges in C as in the classic min-cut problem). Compute, for every point t on the vulnerable plane A, the size of the minimum geometric cut at t. The size of the minimum geometric cut at a point t is the *critical radius* for opens at t.

In the following section, we formulate the Voronoi diagram for opens as a special higher order Voronoi diagram of elements in $core(A)$.

5 A higher order Voronoi digram modeling opens

Let $\mathcal{V}(A)$ denote the (weighted) Voronoi diagram of $core(A)$, the set of all the core elements of nets on layer A. If there were no loops associated with layer A then $\mathcal{V}(A)$ would provide the opens Voronoi diagram on A and $core(A)$ would be the set of all critical generators. $\mathcal{V}(A)$ for Manhattan layouts has been defined in detail in [16]. Fig. 7 illustrates $\mathcal{V}(A)$ for the net graph of Fig. 3. The arrows in Fig. 7 illustrate several minimal radii of defects that break a wire segment. Given a point t in the region of generator s, $t \in reg(s)$, $d_w(t,s)$ gives the radius of the smallest defect centered at t that overlaps the wire segment induced by s. Assuming no loops, $d_w(t,s)$ would be the critical radius of t.

Once loops are taken into consideration, only bridges, articulation and terminal points, among the elements of $core(A)$, correspond to critical generators. Let us augment $\mathcal{V}(A)$ with information reflecting critical generators. In particular, regions of critical generators, that is, regions of first order generators, are colored red. In Fig. 8 red regions are shown shaded and critical generators are shown thickened. The critical radius of point t in a red region $reg(s)$ of owner s is $r_c(t) = d_w(t,s)$.

Let us now define the order-k Voronoi diagram on layer A, denoted as $\mathcal{V}^k(A)$. For $k = 1$, $\mathcal{V}^k(A) = \mathcal{V}(A)$. Following the standard definition of higher order Voronoi diagrams, a region of $\mathcal{V}^k(A)$ corresponds to a maximal locus of points with the same k nearest neighbors among the core elements in $core(A)$. The open portion of a core segment and its two endpoints count as different entities. A kth order Voronoi region belongs to a k-tuple C, $k > 1$, representing the k nearest neighbors to any point in the region of C, denoted as $reg(C)$. The region of C is further subdivided into finer subregions by the farthest Voronoi diagram of C, denoted $V_f(C)$. For any point t in the region of C, $d(t,C) = \max\{d(t,c), \forall c \in C\}$.

In order to appropriately model opens we slightly modify the above standard definition and in certain cases, such as red regions, we allow fewer than k elements to own a Voronoi region of order k. Formally, a red region corresponds to a maximal locus of points with the same r, $1 \le r \le k$, 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

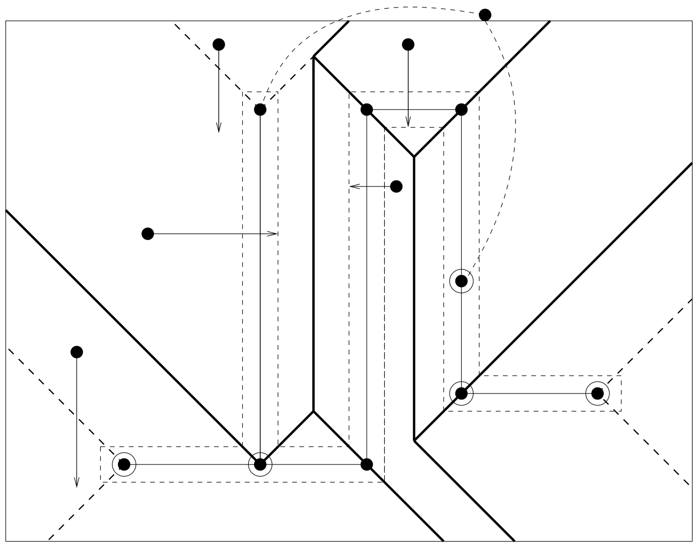


Figure 7: The L_∞ Voronoi diagram of $core(A)$, $\mathcal{V}(A)$, on layer A.

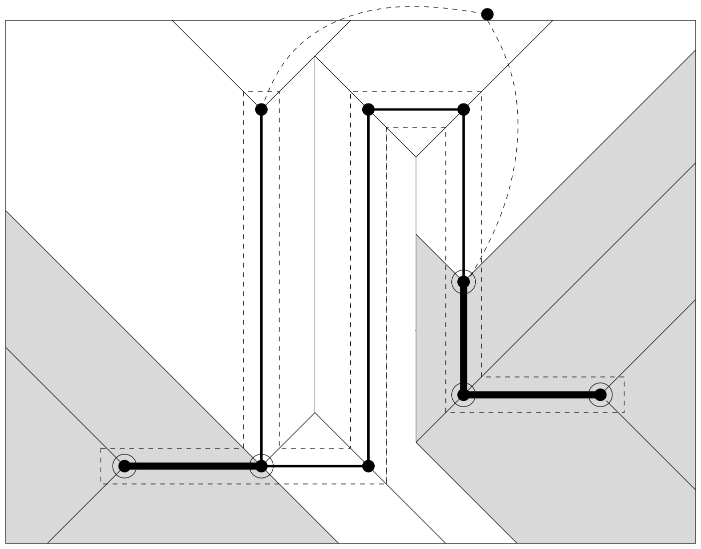


Figure 8: The first order opens Voronoi diagram on layer A, $\mathcal{V}^1(A)$.

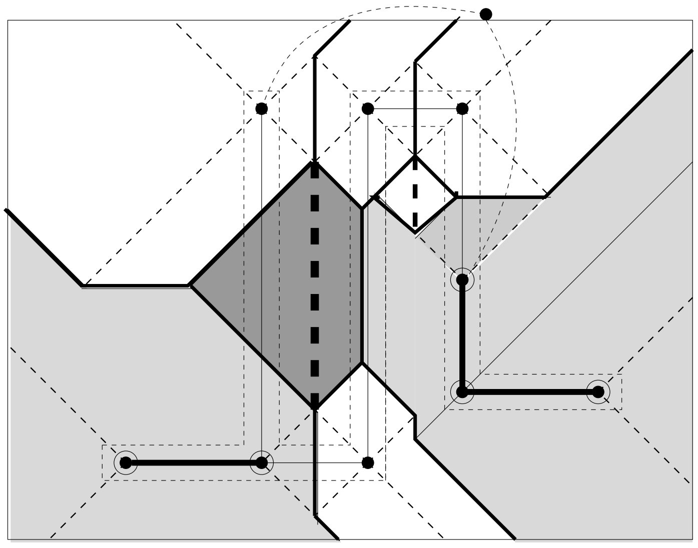


Figure 9: The 2nd order opens Voronoi diagram, $\mathcal{V}^2(A)$.

N. Another deviation from the standard definition regards the open portions of core segments. Any time a core segment s and one of its endpoints p participate in the same set C of nearest neighbors, s is discarded from C. This is because $d(t,p) \geq d(t,s)$ for any point $t \in reg(C)$. Intuitively, a defect that destroys a core endpoint automatically destroys also all incident core segments but not vice versa. In the following, the term *kth order Voronoi diagram* will imply the above modified version of the diagram.

Figs. 9 and 10 illustrate $\mathcal{V}^2(A)$ and $\mathcal{V}^3(A)$ respectively for the net of our example. kth order Voronoi regions are illustrated in solid lines. Red regions are illustrated shaded. The thick dashed lines indicate the farthest Voronoi diagram subdividing a kth order region. In a red region, the thick dashed lines (excluding standard 45°s) correspond to critical generators. All critical generators are indicated thickened: solid ones are first order generators and dashed ones in red regions are higher order generators. All thin dashed lines in Figs. 8, 9, and 10 can be ignored. Note that due to our conventions, the Voronoi region of any core endpoint p in $\mathcal{V}^1(A)$ remains present in $\mathcal{V}^2(A)$ and expands into the regions of its incident core segments. The darker shaded region in $\mathcal{V}^2(A)$ shows the red region of a pair of core elements.

In L_∞, Voronoi subdivisions are not unique but they depend on the conventions used on how to distribute equidistant regions from elements that lie on the same axis parallel lines. For critical area calculations the numerical result will remain the same no matter how equidistant regions get distributed. However, conventions regarding equidistant regions may have an effect on number of iterations and on the shape of the resulting subdivision. We adopt the convention that critical generators get priority over non-critical ones and any region equidistant from a critical and a non-critical generator it is assigned to the critical one and it is colored red.

It is now easy to see that the opens Voronoi diagram on layer A corresponds to the minimum order m Voronoi diagram of $core(A)$, denoted $\mathcal{V}^m(A)$, such that all regions are colored red. Figure 11 illustrates the opens Voronoi diagram, for our example; arrows indicate the critical radius of several points; all critical generators are indicated in thick solid lines. We thus, conclude the following.

Theorem 2 *The Voronoi diagram for opens on layer A is the minimum order m Voronoi diagram of* $core(A)$, $\mathcal{V}^m(A), m \geq 1$, *such that all regions of* $\mathcal{V}^m(A)$ *are colored red. Any region* $reg(H)$, *where* $|H| > 1$, *is subdivided into finer regions by* $V_f(H)$, *the farthest Voronoi diagram of* H. *The critical radius for any point* t *in* $reg(H)$ *is* $r_c(t) = d_w(t,H) = \max\{d_w(t,h), h \in H\}$, *i.e.,* $r_c(t) = d(t,h)$, *where* t *belongs in the subregion of* h *in* $V_f(H)$.

Proof. Let H be a tuple of core elements, $|H| \geq 1$, owning a region of $\mathcal{V}^m(A)$. By definition of a red region, H corresponds to a cut of a biconnected component of $G(N,A)$ for some net N. For any point t in $reg(H)$, H is the nearest cut to t. That is, $d_w(t,H) \leq d_w(t,C)$ for any other cut C on layer A, where $d_w(t,H) = \max\{d_w(t,h), h \in H\}$. Thus, the critical radius at t must be $r_c(t) = d_w(t,H)$. If $H > 1$ then let h be the element of H farthest from t, i.e., $t \in reg(h)$ in $V_h(H)$. Then $r_c(t) = d_w(t,H) = d(t,h)$. □

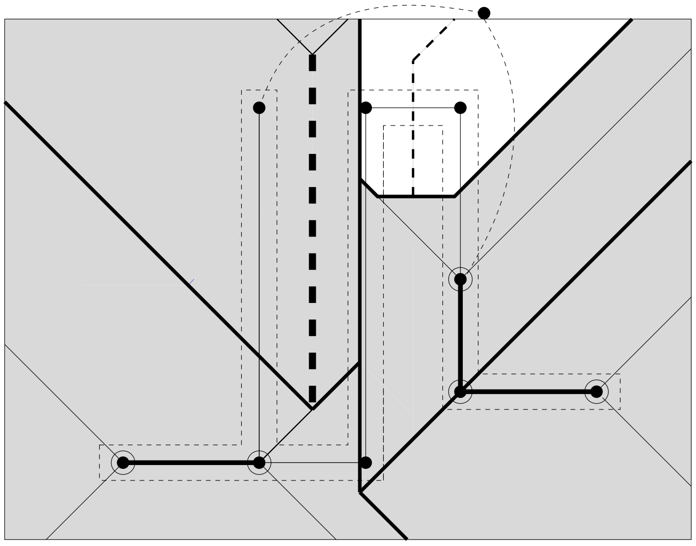


Figure 10: The 3rd order opens Voronoi diagram ,$\mathcal{V}^3(A)$.

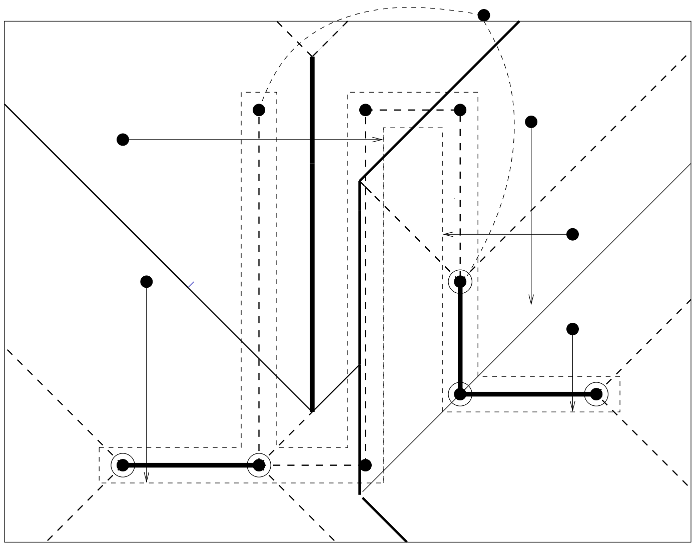


Figure 11: The Voronoi diagram for open faults on layer A.

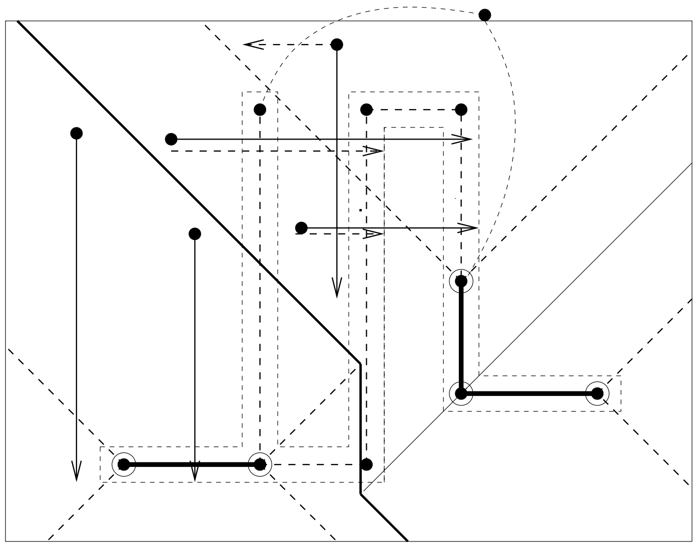


Figure 12: $\mathcal{V}(G_1(A))$ as an approximate opens Voronoi diagram under the (false) assumption that all loops are immune to open faults.

Corollary 2 *The higher order critical generators on layer A are exactly the farthest Voronoi edges and vertices, excluding the standard-45° Voronoi edges, constituting the farthest Voronoi subdivisions in the interior of each region in $\mathcal{V}^m(A)$. All higher order critical generators are encoded in the graph structure of $\mathcal{V}^k(A)$, for some k, $1 \leq k < m$.*

Let $G(A)$ denote the set of all critical generators on layer A including first order and higher order generators. By Theorems 1 and 2, $\mathcal{V}(G(A)) = \mathcal{V}^m(A)$. We classify higher order critical generators according to the minimum order-k Voronoi diagram they first appear in. In particular, higher order generators encoded in $\mathcal{V}^k(A)$ are classified as $(k+1)$-*order generators* and they are denoted as $G_{k+1}(A), 1 \leq k < m$. $G(A) = \cup_{1\leq i\leq m} G_i(A)$.

Given any subset $G'(A)$ of the set $G(A)$ of critical generators, the (weighted) Voronoi diagram of $G'(A)$, $\mathcal{V}(G'(A))$, can be used as an approximation to $\mathcal{V}(G(A))$. Clearly, the more critical generators included in $G'(A)$, the more accurate the result. In practice, we can derive $G'(A)$ as $\cup_{1\leq i\leq k} G_i(A)$, including all ith order generators up to a small constant k. Although there is no guarantee that all critical generators will be discovered in this manner, the result should be sufficient for all practical purposes. Note that the significance of critical generators reduces drastically with the increase in their order.

Corollary 3 *Let $G'(A) = \cup_{1\leq i\leq k} G_i(A)$ be a subset of critical generators including all generators up to order k for a given constant k. The (weighted) Voronoi diagram of $G'(A)$, $\mathcal{V}(G'(A))$, can serve as an approximation to the opens Voronoi diagram $\mathcal{V}^m(A)$. If $G'(A) = G(A)$ then the two diagrams are equivalent.*

Figure 12 illustrates the (weighted) Voronoi diagram of $G_1(A)$ as a rough approximation to the opens Voronoi diagram, $\mathcal{V}(G(A))$. $\mathcal{V}(G_1(A))$ reveals critical radii for opens under the (false) assumption that all loops are immune to open faults. In Figure 12, solid arrows indicate selected critical radii as derived by $\mathcal{V}(G_1(A))$ while dashed arrows indicate true critical radii. Several critical radii can get overestimated in $\mathcal{V}(G_1(A))$ resulting in underestimating the total critical area for open faults. Thus, $\mathcal{V}(G_1(A))$ is not an accurate approximation to $\mathcal{V}(G(A))$. As k increases, however, $\mathcal{V}(\cup_{1\leq i\leq m} G_i(A))$ converges fast to $\mathcal{V}(G(A))$ (see e.g., Section 7). In our example, $\mathcal{V}(G(A))$ is given by $\mathcal{V}(G_1(A) \cup G_2(A))$ as no loops of high connectivity are present. In general, $\mathcal{V}(G_1(A) \cup G_2(A))$ corresponds to the opens Voronoi diagram under the assumption that loops of connectivity higher than two are immune to open faults.

In the next section we describe the algorithm to compute $\mathcal{V}(G'(A))$ and $\mathcal{V}(G(A))$.

6 Computing the opens Voronoi diagram

In this section we give algorithmic details on how to compute $G_{k+1}(A)$ and $\mathcal{V}^{k+1}(A)$, given $\mathcal{V}^k(A)$, for $1 \leq k < m$. We also discuss how to compute $\mathcal{V}(\cup_{1\leq i\leq m} G_i(A))$ and $\mathcal{V}(G(A))$.

6.1 The iterative process to compute higher order generators and higher order opens Voronoi diagrams.

Let's first discuss how to identify the set $G_{k+1}(A)$ of $(k+1)$-order generators, given $\mathcal{V}^k(A)$, for $k \geq 1$.

Consider a non-red region, $reg(H)$, in $\mathcal{V}^k(A)$ and let g be a non-red Voronoi edge bounding $reg(H)$. We need to determine whether g is a critical generator, i.e., whether $g \in G_{k+1}(A)$. Given our conventions, H is an r-tuple, $r \leq k$, of core elements in $core(A)$, representing the k nearest neighbors of every point in $reg(H)$; for any core endpoint p in H all incident core segments have been excluded from H. Let $b(h, j), h \in H, j \in core(A)$, be the bisector inducing g and let B be the biconnected component where h belongs to. Since $reg(H)$ has not been colored red, H is not a cut, and thus, $B - H$ must be connected. We have the following observation: Voronoi edge g corresponds to a critical generator if and only if $H \cup \{j\}$ corresponds to a *cut*; that is, if and only if removing $H \cup \{j\}$ from B disconnects B leaving articulation points in at least two sides.

To determine if g is a critical generator we need to pose a connectivity query to B after removing $H \cup \{j\}$. To perform connectivity queries efficiently we can use the fully dynamic connectivity data structures of [9] that support edge insertion and deletions in $O(log^2 n)$ time, while they can answer connectivity queries fast. For simplicity in our implementation, however, we did not employ any dynamic connectivity data structures; instead we used a very simple (almost brute force) algorithm as follows: Remove the elements of H from B and determine new non-trivial bridges, articulation points and biconnected components of $B - H$. $H \cup \{c\}, c \in core(A)$, constitutes a cut for B if and only if c is a new non-trivial bridge or articulation point of $B - H$. That is, $g \in G_{k+1}(A)$ if and only if j is a new bridge or articulation point of $B - H$. Generator g gets associated with the tuple of core elements $J = H \cup \{j\}$ (resp. $J = H \cup \{j\} - \{s\}$ in case j is a core endpoint incident to core segment $s \in H$).

The above process can be considerably simplified in the case where the biconnected component B is a simple cycle. In this case a simple coloring scheme in the DFS tree of B can efficiently identify all cuts of B that may be associated with a second order generator. Biconnected components forming simple cycles appear often in practice, thus, simplifying and speeding up the process when encountered. The time complexity of determining $G_{k+1}(A)$ given $\mathcal{V}^k(A)$ is summarized in the following lemma. Note that the size of $\mathcal{V}^k(A)$ is $O(k(n-k))$ (see [13]).

Lemma 2 *The $(k+1)$-order generators can be determined from $\mathcal{V}^k(A)$ in time $O(kn\log^2 n)$ using the dynamic connectivity data structures of [9] or in time $O(kn^2)$ using the simple algorithm presented above. In case of biconnected components forming simple cycles, second order generators can be determined from $\mathcal{V}(A)$ in linear time.*

Let us now discuss how to obtain $\mathcal{V}^{k+1}(A)$ from $\mathcal{V}^k(A)$ for $k \geq 1$. The following is an adaptation of the iterative process to compute higher order Voronoi diagrams of points [13], to the case of (weighted segments).

Let $reg(H)$ be a non-red region of $\mathcal{V}^k(A)$. H is an r-tuple, $r \leq k$, of core elements representing the k nearest neighbors of $reg(H)$. Let $N(H)$ denote the set of all core elements that induce a bisector bounding $reg(H)$ in $\mathcal{V}^k(A)$. Compute the (weighted) L_∞ Voronoi diagram of $N(H)$ and truncate it within the interior of $reg(H)$; this gives the $(k+1)$-order subdivision within $reg(H)$. Each $(k+1)$-order subregion of $reg(H)$ is attributed to a tuple $J = H \cup \{c\}$ for some core element c in $N(H)$. In case c is a core point incident to a core segment s in H then J simplifies to $J = H - \{s\} \cup \{c\}$. In case c is part of a cut C owning a neighboring red region of $\mathcal{V}^k(A)$ then the subregion of J gets colored red and gets as owner the cut C. Once the $(k+1)$-order subdivision has been performed within all neighboring non-red regions, merge any incident $(k+1)$-order subregions that belong to the same tuple of owners J into a maximal $(k+1)$-order region, $reg(J)$. The edges of $\mathcal{V}^k(A)$ included within $reg(J)$ constitute the farthest Voronoi diagram of J and they can remain in $\mathcal{V}^{k+1}(A)$ providing the finer subdivision of $reg(J)$ into farthest regions. Note that all $(k+1)$-order red subregions get merged with the neighboring red regions of $\mathcal{V}^k(A)$ into maximal red regions of $\mathcal{V}^{k+1}(A)$. Using established bounds for higher order Voronoi diagrams of points (see e.g. [13]) we conclude the following.

Lemma 3 *$\mathcal{V}^{k+1}(A)$ can be computed from $\mathcal{V}^k(A)$ in time $O(k(n-k)\log n)$, plus the time $T(k,n)$ to determine the $(k+1)$-order generators, where $T(k,n)$ is as given in Lemma 2.*

6.2 Computing the opens Voronoi diagram from critical generators.

The iterative process of Section 6.1 can continue until all regions are colored red and the complete opens Voronoi diagram is guaranteed to be available. In practice, however, this would be inefficient. Note that the iterative process may continue for several rounds without any new critical generators being identified, only the regions of existing critical generators keep enlarging into neighboring non-red regions. Note also that as the number of iterations k increases, the weight of order-k critical generators (if any) increases as well and their effect on total critical area gets reduced.

In practice, we can restrict the number of iterations to a small predetermined constant k, or to a small number determined adaptively, and compute a sufficient set of critical generators $G'(A) = \cup_{1\leq i\leq k} G_i(A)$. We can then use Theorem 1 to report $\mathcal{V}(G'(A))$ as the opens Voronoi diagram. Thus, the overall algorithm can be broken into two independent parts:

- Part I: Compute the set of critical generators $G'(A) = \cup_{1\leq i\leq k} G_i(A)$, up to a given (or adaptively determined) order k.
- Part II: Compute the weighted Voronoi diagram of $G'(A)$, $\mathcal{V}(G'(A))$.

Part I can be performed using the iterative process of Section 6.1. The maximum order k can be restricted to a small value k, $k = 2,3,4$. Experimental results in Section 7 suggest that even $k = 2$ is adequate in most cases and no $k > 4$ is ever needed. Alternatively, k can be determined adaptively, e.g., k can be set to the first round such that no new critical generators are determined. Part II can be performed using the same plane sweep algorithm as computing $\mathcal{V}(A)$, given that all generators produced in Part I are critical. Critical generators have similar properties to the elements of $core(A)$, and thus, the same plane sweep algorithm can also be used to compute $\mathcal{V}(G'(A))$ (see [16, 20]).

The computations of Parts I and II can be synchronized as there is no need for Part I to be complete in order to start the computation of Part II; once a generator is discovered in Part I, it can be immediately scheduled to be processed in Part II. Details on how to achieve the synchronization are given in the next section. In the following, we review the basic concepts of the plane sweep construction of Voronoi diagrams. For more details see [16, 20].

Imagine a vertical scan line L sweeping layer A from left to right. Associated with a plane-sweep algorithm there are two major components: a *sweep-line status*, $\mathcal{T}$, maintaining the status of the sweeping process, and an *event list*, Q, containing the *events* where the combinatorial structure of the sweep-line status may change. Q is ordered in increasing order of event *priority*. The *priority* of a generator point p is given by the rightmost coordinate of a square centered at p having radius $w(p)$, i.e., $priority(p) = x_p + w(p)$, where x_p is the x-coordinate of p. The *priority* of any Voronoi point p is defined in the same way, where $w(p)$ is the (weighted) distance of p from its defining elements. Throughout the sweeping process, a partial Voronoi diagram so far of all generators having priority less or equal to the current position of the scan line, including the scan line, is maintained. The collection of Voronoi edges (portions of bisectors) bounding the Voronoi cell of the scan line is called the *wavefront*. As the scan line moves to the right, the wavefront as well as the endpoints of incident bisectors also move to the right. Any Voronoi point enters the wavefront at the time of its priority. The combinatorial structure of the wavefront is maintained in the sweep-line status, $\mathcal{T}$, which gets implemented as a *height-balanced* tree (see e.g. [6]). The events where the combinatorial structure of the wavefront may change are maintained in the event list, Q, which is implemented as a *priority queue*. At every event, the wavefront gets updated, and new events may get generated; any portion of the wavefront that gets finalized it joins the Voronoi diagram computed so far. Once the handling of an event is complete, the scanline proceeds to the next event in the priority Q. When all events are processed the construction of the Voronoi diagram is complete. For more details see [20, 22].

6.3 Synchronizing plane sweeps and practical considerations

An important advantage of the plane sweep approach to the construction of Voronoi diagrams and the extraction of critical area has been locality: The Voronoi diagram of a layer need never be kept in memory in order to perform critical area extraction; once an appropriate Voronoi region is computed critical area computation is directly performed in that region and the Voronoi region can be discarded. Recall that critical area computation within a Voronoi cell, in the case of the $D(r) = r_0/r^3$ distribution, corresponds to an addition of simple formulas derived from Voronoi edges, see [16, 22]. We would like to synchronize the plane sweeps of Parts I and II so that the locality property is maintained. We first discuss the simpler case where the opens Voronoi diagram is reported as $\mathcal{V}(G_1(A) \cup G_2(A))$, considering only first and second order generators. This simpler case is important as restricting k to $k = 2$ seems to be adequate in practice. We then generalize to other small values of k.

Let L_I and L_{II} denote the scanlines for Parts I and II respectively. Both L_I and L_{II} work as described in [16, 20] with the difference that 2nd order generators are not known a priori but they are determined on the fly by L_I, at the time of their minimum priority. At every event of L_I where a new Voronoi edge or a new bisector touching the wavefront, say g, is determined, we can check whether g corresponds to a cut generator as described in Section 6.1. If so, a new generator event is created for L_{II} having as priority the current position of L_I. Note that red and standard-45 Voronoi edges cannot be critical generators and thus they need never be checked. L_I executes the plane sweep construction of $\mathcal{V}(A)$, however, it only needs to maintain the wavefront; every time an element of $\mathcal{V}(A)$ leaves the wavefront it can be directly discarded. The goal of L_I is to discover events of 2nd order generators and feed them to L_{II}.

L_{II} computes $V(G_1(A) \cup G_2(A))$ following the algorithm of [16,20] with the difference that it receives events regarding second order generators from L_I. L_{II} need never keep in memory the entire $V(G_1(A) \cup G_2(A))$; once a Voronoi cell of $V(G_1(A) \cup G_2(A))$ leaves the wavefront, critical area extraction can be directly performed in that cell (see [16]) and the Voronoi cell can then be discarded. L_{II} maintains Voronoi cells while they are incident to the wavefront, preserving the locality property.

The synchronization of the two sweeps for Parts I and II can generalize to $k > 2$, if desirable. The generalization is practical for small values of $k, k \leq 4$. For any higher value we would recommend a slightly different approach to be described in Section 6.4. In practice, it is highly unlikely that any larger k would ever be needed. Synchronization is desirable for applying the method to large blocks of layout.

In more detail, the generalization to $k > 2$ can be done as follows. L_I computes $\mathcal{V}(A)$ and it maintains the wavefront of $\mathcal{V}(A)$ and all Voronoi cells incident to it. Once a Voronoi cell V leaves the wavefront, the higher order Voronoi diagram construction can start within V. First, the edges along the boundary of V need to be checked to decide whether they correspond to second order generators. If any of them does, a generator event is created and fed to L_{II}. Then, the 2nd order Voronoi subdivision is performed within V (as described in Section 6.1). Every second order subregion in V need to be merged with a neighboring second order subregion, V_2, included in a Voronoi cell neighboring V, if available. If V_2 is available then the two subregions get merged into a new second order Voronoi region; it is then decided whether or not the new second order region is colored red. If the neighboring second order subregion V_2 is not available then the completion of the full 2nd order Voronoi region will have to wait. To make sure that L_{II} cannot advance beyond the priority of any higher order generator induced from V_2, we keep the minimum priority of V_2 into a heap H of restricted priorities for L_{II}; L_{II} can only advance up to the minimum value of H. Every time a 2nd order subregion V_2 gets merged into a 2nd order region, its priority gets deleted from H. Any region determined to be red can be marked as *finished* as it can not contribute in the derivation of new generators. A non-red region of order $(k-1)$ i.e., maximum order, can also be marked *finished*. Once a new 2nd order Voronoi region is created the process can repeat in deriving the 3rd order subdivision within, determining new 3rd order generators (if any), merging each subregion (if possible) with neighboring 3rd order Voronoi subregions if available, and updating the priority heap H with the minimum priority value of each subregion that still has to wait.

L_{II} always advances to the minimum priority event of its event list, after confirming that this is lower than the minimum priority in H. L_{II} may have to wait until the processing of L_I increases the minimum priority value in H to guarantee that events are always processed in increasing priority value. Once a Voronoi cell and all its neighbors have been marked as *finished* the cell can be discarded.

6.4 Original implementation

Our original implementation, whose experimental results are reported in Section 7, used a slightly different approach in order to guarantee accuracy while the locality property was preserved. Namely, the iterative process of Section 6.1 was applied to each biconnected component independently. The advantage of considering each biconnected component independently is locality as well as the ability to run the process on each individual component to completion and thus, guarantee the accuracy. The disadvantage, however, is that the generators produced in this manner need not all be critical. Including non critical generators in the set of generators obtained in Part I, denoted $G''(A)$, complicates the algorithm of Part II to compute the weighted Voronoi diagram of $G''(A)$. For modifications of the plane sweep in the presence on non-critical generators see [16, 17]. $\mathcal{V}(G''(A))$ corresponds to the *Hausdorff Voronoi diagram* of all cuts on layer A. For information on Hausdorff Voronoi diagrams the interested reader is referred to [17, 18, 21, 31].

After obtaining the experimental results of Section 7 using the provided guarantees of accuracy, we verified that small values of k are adequate for accurate critical area extraction. Given the experimental results, in practice, we recommend the approach described in Section6.3 to compute either $\mathcal{V}(G_1(A) \cup G_2(A))$ or $\mathcal{V}(G_1(A) \cup G_2(A) \cup G_3(A))$ as the opens Voronoi diagram and extract critical area information.

7 Experimental results

The algorithms presented in this paper have been implemented as part of the net-aware opens capability of the IBM Voronoi Critical Area Analysis (CAA) tool [28]. The original tool is currently distributed by Cadence [5] providing critical area analysis for shorts, opens, via-blocks, and combination faults, via Voronoi diagrams. For results on the use of an early version of the tool at IBM, without the net-aware opens capability, see [14].

We ran the net aware capability of the IBM Voronoi CAA tool on a number of blocks from IBM 65nm and 45nm silicon-on-insulator (SOI) microprocessor designs. The sizes of some blocks are summarized in Table 1 given in square microns and number of transistors. Table 2 summarizes the results of the runs and reports the Probability of Fault (POF) as computed for an increasing number of iterations k to produce higher order generators for opens. For each $k = 1, 2, 3, \ldots$ the POF value reported is determined by $\mathcal{V}(G'(A))$, where $G'(A) = \cup_{1 \le i \le k} G_i(A)$, using the formulas to compute critical area reported in [16, 20], assuming square defects following the $D(r) = r_0/r^3$ defect distribution. As expected the POF converges very fast to a final value that remains the same although k is allowed to increase up to a large value. This final value is the POF as obtained by the opens Voronoi diagram, $\mathcal{V}(G(A))$. To verify accuracy, our experiments were run allowing much larger values of k than those reported in Table 2, however no further improvement to POF was reported allowing us to conclude that the final opens Voronoi diagram $\mathcal{V}(G(A))$ was obtained at rather small values of k. The algorithm followed the variant reported in Section 6.4.

Given the experimental results in Table 2 we observe that there is hardly any need to compute kth order generators for opens for any $k > 4$. Only in one case (see block F2-M2) the total POF kept on slightly increasing until iteration $k = 8$, which implied that loops of high connectivity were found vulnerable to open faults contributing small amounts to total critical area as generators of higher order k kept on being discovered. Even in this case, however, the important increase happens early for $k \le 3$. Given the experimental results, we recommend to compute Critical Area using the simplified opens Voronoi diagram obtained by $\mathcal{V}(G_1(A) \cup G_2(A))$ that can be derived in a rather simple manner, avoiding any iteration, as explained in Section 6.3. Alternatively, $\mathcal{V}(G_1(A) \cup G_2(A) \cup G_3(A))$ seems accurate enough for most practical purposes. The plain numeric values of POF as reported Table 2 may not seem particularly informative stand alone. However, the main importance of the CAA tool lies in the ability to perform comparisons in a reliable way rather than the absolute values of the POF obtained.

Block ID	square microns	# of transistors
B1	13631	22608
B3	9661	17935
B4	4161	10988
S1	5639	11482
S2	13926	30360
F1	30470	39923
F2	22550	34467

Table 1: Sample block sizes from Table 2 in square microns and number of transistors.

Figure 13 illustrates charts of some sample results of Table 2. Each chart plots the Probability of Fail (POF) for opens on a given layer of a block (M1, M2, M3, PC) given on the Y-axis, versus the maximum number k of higher order generators allowed, given in the X-axis. The POF is derived from $\mathcal{V}(G'(A))$, where $G'(A) = \cup_{1 \le i \le k} G_i(A)$, for any layer $A = M1, M2, M3, PC$. Note that the largest improvement typically takes place as k increases from 1 to 2 and in some cases from 2 to 3. Any value for k above 4 is hardly ever needed.

8 Conclusion

In this paper we modeled the critical area computation problem for open faults in the presence of loops and redundant interconnects, and reduced the problem into generalizations of higher order Voronoi diagrams of segments. The approach extends the Voronoi based method for critical area extraction with the ability to accurately compute critical area in a net-aware fashion even in the presence of multilayer loops.

In the process we introduced the *geometric min cut problem*, a geometric version of the classic min-cut problems in graphs. We also generalized the iterative approach to compute higher order Voronoi diagrams in the case of line segments and introduced special features to adequately model open faults. Surprisingly

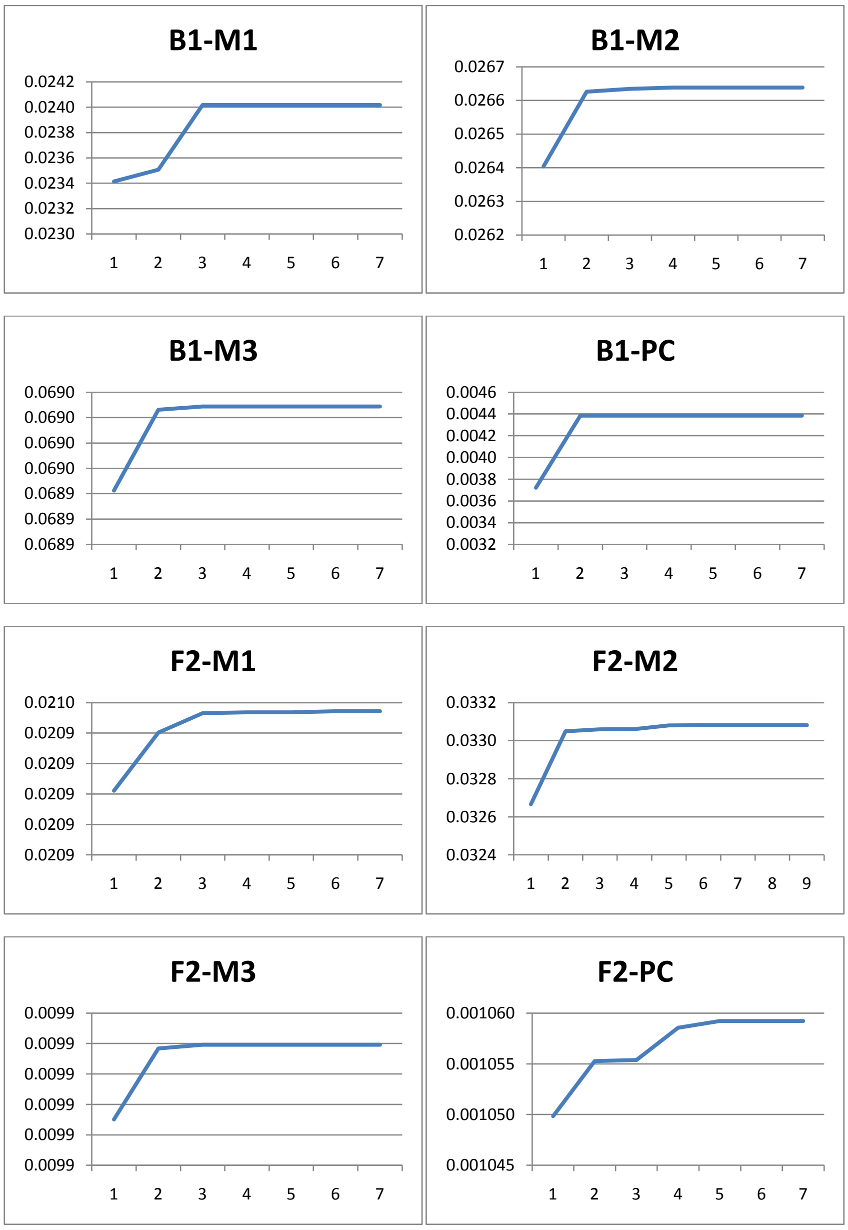


Figure 13: Plotted results of some of the entries from Table 2: POF as a function of maximum number of iterations k to derive $G'(A) = \cup_{1 \le i \le k} G_i(A)$. Each plot shows the POF for a particular layer of a block; e.g., F2-x, where $x = M1, M2, M3, PC$. The POF converges fast to its final value.

Block ID	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$
45nm SOI								
B1-M1	0.02341480	0.02350720	0.02401760	0.02401770	0.02401770	0.02401770	0.02401770	
B1-M2	0.02640390	0.02662630	0.02663450	0.02663830	0.02663830	0.02663830	0.02663830	
B1-M3	0.06894250	0.06900620	0.06900880	0.06900880	0.06900880	0.06900880	0.06900880	
B1-PC	0.00372151	0.00438500	0.00438500	0.00438500	0.00438500	0.00438500	0.00438500	
B2-M1	0.00559925	0.00562743	0.00563807	0.00563856	0.00563905	0.00563905	0.00563905	
B2-M2	0.00945846	0.00950235	0.00950272	0.00950272	0.00950272	0.00950272	0.00950272	
B2-M3	N/A	0.01757380	0.01757380	0.01757380	0.01757380	0.01757380	0.01757380	
B2-PC	0.00278295	0.00278295	0.00278295	0.00278295	0.00278295	0.00278295	0.00278295	
B3-M1	0.02342090	0.02355120	0.02426700	0.02426700	0.02426700	0.02426700	0.02426700	
B3-M2	0.02686700	0.02714910	0.02717070	0.02717400	0.02717400	0.02717400	0.02717400	
B3-M3	0.07530690	0.07539750	0.07540190	0.07540190	0.07540190	0.07540190	0.07540190	
B3-PC	0.00314780	0.00360734	0.00360734	0.00360734	0.00360734	0.00360734	0.00360734	
B4-M1	0.06903250	0.06933660	0.06944170	0.06944170	0.06944170	0.06944170	0.06944170	
B4-M2	0.07679910	0.07752170	0.07753360	0.07753860	0.07753860	0.07753860	0.07753860	
B4-M3	0.07013570	0.07032030	0.07032270	0.07032270	0.07032270	0.07032270	0.07032270	
B4-PC	0.00494624	0.00556126	0.00556126	0.00556126	0.00556126	0.00556126	0.00556126	
S1-M1	0.05102740	0.05150460	0.05221020	0.05221020	0.05221020	0.05221020	0.05221020	
S1-M2	0.08669180	0.08724550	0.08727970	0.08729350	0.08731620	0.08731620	0.08731620	
S1-M3	0.05813100	0.05816990	0.05817110	0.05817110	0.05817150	0.05817150	0.05817150	
S1-PC	0.00165505	0.00165505	0.00165505	0.00165505	0.00165505	0.00165505	0.00165505	
S2-M1	0.04333750	0.04370110	0.04404340	0.04405340	0.04405340	0.04405340	0.04405340	
S2-M2	0.07599400	0.07686370	0.07704780	0.07705310	0.07705310	0.07705310	0.07705310	
S2-M3	0.06442370	0.06452310	0.06452380	0.06452490	0.06452490	0.06452490	0.06452490	
S2-PC	0.00170875	0.00170875	0.00170875	0.00170875	0.00170875	0.00170875	0.00170875	
65nm SOI								
F1-M1	0.03083600	0.03086650	0.03088090	0.03088090	0.03088090	0.03088090	0.03088090	
F1-M2	0.02325990	0.02416790	0.02430430	0.02430440	0.02430440	0.02430440	0.02430440	
F1-M3	0.02496010	0.02504540	0.02504570	0.02504570	0.02504570	0.02504570	0.02504570	
F1-PC	0.00225416	0.00225416	0.00225416	0.00225416	0.00225416	0.00225416	0.00225416	
F2-M1	0.02090210	0.02094040	0.02095310	0.02095370	0.02095370	0.02095430	0.02095430	
F2-M2	0.03266590	0.03304950	0.03305970	0.03306080	0.03308010	0.03308150	0.03308180	0.03308200
F2-M3	0.00986754	0.00987918	0.00987979	0.00987979	0.00987979	0.00987979	0.00987979	
F2-PC	0.00104983	0.00105527	0.00105537	0.00105856	0.00105922	0.00105922	0.00105922	
F3-M1	0.00206703	0.00207091	0.00208208	0.00208208	0.00208208	0.00208208	0.00208208	
F3-M2	0.00342026	0.00342420	0.00342420	0.00342420	0.00342420	0.00342420	0.00342420	
F3-M3	0.00017472	0.00017472	0.00017472	0.00017472	0.00017472	0.00017472	0.00017472	
F3-PC	0.00211915	0.00211915	0.00242553	0.00242553	0.00242553	0.00242553	0.00242553	
F4-M1	0.01103590	0.01104400	0.01105820	0.01105820	0.01105820	0.01105820	0.01105820	
F4-M2	0.02046730	0.02059360	0.02059780	0.02059780	0.02059780	0.02059780	0.02059780	
F4-M3	0.01085790	0.01086880	0.01086880	0.01086880	0.01086880	0.01086880	0.01086880	
F4-PC	0.00617254	0.00617254	0.00617254	0.00617254	0.00617254	0.00617254	0.00617254	

Table 2: Probability of Fault (POF) for opens versus maximum order k of critical generators in $\mathcal{V}(G'(A))$ on various IBM microprocessor blocks.

higher order Voronoi diagrams of line segments had not been addressed in the computational geometry literature.

Our early algorithms have been integrated in the *IBM Voronoi Critical Area Analysis* (CAA) tool that is currently used in production mode by IBM manufacturing. Using the net aware open capability of this tool we provided experimental results that verify the ability to simplify the method in practice without compromising the level of accuracy. For completeness we presented the full method that can guarantee the accuracy but we also presented several practical simplifications.

In summary, the Voronoi method to extract critical area for various types of faults computes the entire critical area integral for all possible defect sizes in an analytical way, which can be applied while an appropriate Voronoi subdivision is obtained. It can be used stand alone for fast and accurate critical area extraction on rather large layout blocks or it can be combined with layout sampling techniques (see [1, 4]) for fast critical area estimation at the chip level. The Voronoi approach to critical area extraction has been developed into a successful industrial tool that is currently used extensively in production mode by IBM Microelectronics for the prediction of yield.

Acknowledgment

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 entire 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. The net-aware critical area capability described in this paper was implemented by Mervyn Y. Tan. Mervyn is greatly acknowledged for his outstanding contribution to the entire development of the IBM Voronoi CAA tool. I would like to particularly thank Jason Hibbeler for his help in setting up the experiments whose results are reported in this paper.

References

[1] G. A. Allan. Yield prediction by sampling IC layout. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(3):359–371, 2000.

[2] G. A. Allan and A. Walton. Efficient extra material critical area algorithms. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18(10):1480–1486, 1999.

[3] F. Aurenhammer, R. Drysdale, and H. Krasser. Farthest line segment Voronoi diagrams. *Information Processing Letters*, 100:220–225, 2006.

[4] S. C. Braasch, J. Hibbeler, R. N. Kanj, D. Maynard, S. Nassif, and E. Papadopoulou. Method and system for analyzing an integrated circuit based on sample windows selected using an open deterministic sequencing technique. Patent application US20090031263, Filed, April 2007.

[5] S. C. Braasch, J. Hibbeler, D. Maynard, M. Koshy, R. Ruehl, and D. White. Model-based verification and analysis for 65/45nm physical design. CDNLive!, September 2007.

[6] M. de Berg, O. Schwarzkopf, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications.* Springer-Verlag, 2nd edition, 2000.

[7] J. P. de Gyvez and C. Di. IC defect sensitivity for footprint-type spot defects. *IEEE Trans. on Computer-Aided Design*, 11(5):638–658, 1992.

[8] A. Ferris-Prabhu. Defect size variations and their effect on the critical area of VLSI devices. *IEEE J. of Solid State Circuits*, 20(4):878–880, Aug. 1985.

[9] J. Holm, K. Lichtenberg, and M. Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *J. ACM*, 48(4):723–760, 2001.

[10] J. Hopcroft and R. Tarjan. Efficient algorithms for graph manipulation. *Comm. ACM*, 16(6):372–378, 1973.

[11] A. B. Kahng, B. Liu, and I. I. Mandoiu. Non-tree routing for reliability and yield improvement. *IEEE Trans. on Comp. Aided Design of Integrated Circuits and Systems*, 23(1):148 – 156, 2004.

[12] I. Koren. *Yield Modeling and defect Tolerance in VLSI circuits*, chapter The effect of scaling on the yield of VLSI circuits, pages 91–99. Adam-Hilger Ltd., 1988.

[13] D. T. Lee. On k-nearest neighbor Voronoi diagrams in the plane. *IEEE Trans. Comput.*, C-31(6):478–487, June 1982.

[14] D. N. Maynard and J. D. Hibbeler. Measurement and reduction of critical area using Voronoi diagrams. In *Advanced Semiconductor Manufacturing IEEE Conference and Workshop*, 2005.

[15] K. Mehlhorn, S. Meiser, and R. Rasch. Furthest site abstract voronoi diagrams. *Internat. J. Comput. Geom. Appl.*, 11(6):583–616, 2001.

[16] E. Papadopoulou. Critical area computation for missing material defects in VLSI circuits. *IEEE Transactions on Computer-Aided Design*, 20(5):583–597, 2001.

[17] E. Papadopoulou. The Hausdorff Voronoi diagram of point clusters in the plane. *Algorithmica*, 40:63–82, 2004.

[18] E. Papadopoulou. Higher order Voronoi diagrams of segments for VLSI critical area extraction. In *Proc. 18th International Symposium on Algorithms and Computation*, volume 4835 of *Lecture Notes in Computer Science 4835*, pages 716–727, 2007.

[19] E. Papadopoulou. The higher order Hausdorff Voronoi diagram and VLSI critical area extraction for via-blocks. In *Proc. 5th Int. Symposium on Voronoi diagrams in Science and Engineering*, pages 181–191, 2008.

[20] E. Papadopoulou and D. Lee. The L_∞ Voronoi diagram of segments and VLSI applications. *International Journal of Computational Geometry and Applications*, 11(5):503–528, 2001.

[21] E. Papadopoulou and D. Lee. The Hausdorff Voronoi diagram of polygonal objects: A divide and conquer approach. *International Journal of Computational Geometry and Applications*, 14(6):421–452, 2004.

[22] E. Papadopoulou and D. T. Lee. Critical area computation via Voronoi diagrams. *IEEE Transactions on Computer-Aided Design*, 18(4):463–474, 1999.

[23] W. A. Pleskacz, C. H. Ouyang, and W. Maly. Extraction of critical areas for opens in large VLSI circuits. *IEEE Trans. on Computer-Aided Design*, 18(2):151–162, 1999.

[24] J. S. Rogenski. Extraction of breaks in rectilinear layouts by plane sweeps. Master's thesis, University of California, Santa Cruz, April 1995.

[25] C. H. Stapper. Modeling of defects in integrated circuit photolithographic patterns. *IBM J. Res. Develop.*, 28(4):461–475, 1984.

[26] C. H. Stapper and R. J. Rosner. Integrated circuit yield management and yield analysis: Development and implementation. *IEEE Trans. Semiconductor Manufacturing*, 8(2):95–102, May 1995.

[27] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1:146–160, 1972.

[28] Voronoi CAA: Voronoi Critical Area Analysis. IBM CAD Tool, Department of Electronic Design Automation, IBM Microelectronics Division, Burlington, VT. Initial patents: US6178539, US6317859.

[29] I. A. Wagner and I. Koren. An interactive VLSI CAD tool for yield estimation. *IEEE Trans. on Semiconductor Manufacturing*, 8(2):130–138, 1995.

[30] H. Walker and S. W. Director. VLASIC: A yield simulator for integrated circuits. *IEEE Trans. on Computer-Aided Design*, 5(4):541–556, 1986.

[31] J. Xu, L. Xu, and E. Papadopoulou. Computing the map of geometric minimal cuts. In *Proc. 20th International Symposium on Algorithms and Computation*, volume 5878 of *LNCS*, pages 244–254, 2009.

[32] S. T. Zachariah and S. Chakravarty. Algorithm to extract two-node bridges. *IEEE Transactions on VLSI Systems*, 11(4):741–744, 2003.